# MODELING, LEARNING, AND INFERENCE OF HIGH-DIMENSIONAL ASYNCHRONOUS EVENT DATA

A Thesis
Presented to
The Academic Faculty

by

Nan Du

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computational Science and Engineering

Georgia Institute of Technology
August 2016

# MODELING, LEARNING, AND INFERENCE OF HIGH-DIMENSIONAL ASYNCHRONOUS EVENT DATA

Approved by:


Professor Le Song, Advisor
College of Computing
*Georgia Institute of Technology*

Professor Christos Faloutsos
Department of Computer Science
*Carnegie Mellon University*

Professor Hongyuan Zha
College of Computing
*Georgia Institute of Technology*

Dr. Evgeniy Gabrilovich
Google Research

Professor Yu Hu
Scheller College of Business
*Georgia Institute of Technology*


Date Approved: April 26, 2016

*To my beloved Mom, Dad,*

*Pengpeng, and Ethan.*

# ACKNOWLEDGEMENTS

In people's lives, a few important moments and opportunities may significantly change the trajectory of one's career and life. It is such a great fortune for me to meet my advisor, Le Song, in Georgia Tech at one of these few moments in 2011. I would like to express my deepest gratitude and appreciation to Professor Le Song for his visionary, inspiring, and insightful guidance and support which truly set the course to what is here today. I still remember that five years ago when I first came to Georgia Tech without any machine learning background, it is Le's kind encouragement and patience to help me move to the right track. Without his extraordinary supervision, I could not achieve what I have right now.

My research and growth in machine learning also benefit significantly from interacting with a marvelous group of coauthors and collaborators. I would like to extend my sincere appreciation to each of them: Amr Ahmed, Maria-Florina (Nina) Balcan, Hanjun Dai, Abhimanyu Das, Christos Faloutsos, Mehrdad Farajtabar, Evgeniy Gabrilovich, Yingyu Liang, Niao He, Alexander J. Smola, Le Song, Rakshit Trivedi, Yichen Wang, Hyenkyun Woo, Hongyuan Zha, *etc.* I am especially grateful to Christos for his generous help and advice, and to Hongyuan for his suggestions for career endeavor. I would like to thank Mehrdad for active discussions and close collaborations. I will never forget the days when we sit up all night together to finish each 'deadline' one after another. I have learned a lot from Yingyu Liang and Niao He. Thank you for the unselfish help and insights during our collaboration. I would like to thank Professor Richard Vuduc and Polo Chau who give me great help and ideas of bringing high performance computing and various useful visualization techniques into my research. I also thank Hanjun and Yichen for the timely support whenever I

need help. I have great memories and experiences at Google Research. Thank Amr, Alex, Abhi, and Evgeniy for bringing me to many new fields, which greatly broadens the horizon of my research.

My Ph.D. life can never become more wonderful without my fellow friends and classmates. I am grateful to Bo Xie, Hua Ouyang, Da Kuang, Joonseok Lee, Fuxing Li, Liangda Li, Bo Dai, Marat Dukhan, Shuanghong Yang, Ke Zhou, Elias Khalil, Jiajia Li, Zhichen Xia, Edward Choi, Hang Su, Mohammad Taha Bahadori, Yuyu Zhang for long discussions and funny chats about machine learning, programming, personal life, career path and everything else. Special thanks also go to Mimi Haley and Deanna Richards who timely help me to deal with all kinds of cumbersome forms and signatures during my Ph.D. study.

I would like to express my gratitude to the committee members, Hongyuan Zha, Yu Hu, Christos Faloutsos, and Evgeniy Gabrilovich for their valuable comments and constructive feedbacks.

Finally and even more importantly, I feel deeply indebted to the selfless love and endless support of my parents. I cannot express in words my gratitude to them and to my parents in law for taking care of Ethan by flying back and forth across half of the earth. My wife, Pengpeng, I always thank you for endeavoring with me in Georgia Tech and for believing in me unconditionally. You, Ethan, and our parents are always the most important part of my life. I love you all so much.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# SUMMARY

The increasing availability of temporal-spatial events produced from natural and social systems provides new opportunities and challenges for effective modeling the latent dynamics which inherently govern these seemly "random" data. In our work, we propose a unified probabilistic framework based on multivariate point processes to better predict '*who will do what by when and where*?' in the future. This framework comprises a systematic paradigm for modeling, learning, and making inference of large-scale asynchronous high-dimensional event data. With this common framework, we contribute in the following three aspects:

- **Accurate Modeling.** We first propose non-parametric and topic-modulated multivariate terminating point processes to capture continuous-time heterogeneous information diffusions. We then develop the low-rank Hawkes process to describe the recurrent temporal interactions among different types of entities. We also build a link between the recurrent neural network and the temporal point process to learn a general representation of the influence from the past event history. Finally, we establish a previously unexplored connection between Bayesian Nonparametrics and temporal point processes to jointly model the temporal data and other type of additional information.

- **Efficient Learning.** We develop a robust structure learning algorithm via group lasso, which is able to efficiently uncover sparse heterogeneous interdependent relations specified via vectorized parameters among the dimensions. We also propose an efficient nonnegative matrix rank minimization algorithm, which elegantly inherits the advantages from both the proximal methods and the conditional gradient methods to solve the matrix rank minimization problem

under different constraints. Finally, in the data streaming setting, we develop a Bayesian inference algorithm for inferring latent variables and updating the respective model parameters based on both temporal and textual information, which achieves almost constant processing time per data sample.

- **Scalable Inference.** Another important aspect of our research is to make future predictions by exploiting the learned models. Specifically, based on the terminating processes, we develop the first scalable influence estimation algorithm in continuous-time diffusion networks with provable performance guarantees. Based on the low-rank Hawkes processes, we develop the first time-sensitive recommendation algorithm, which not only can recommend the most relevant item specific to a given moment, but also can predict the next returning time for a user to a designated service. Finally, based on the recurrent point processes, we have derived an analytic solution to shape the overall network activities of users. We show that our method can provide fine-grained control over user activities in a time-sensitive fashion.

**Keywords:** Multivariate point process, Hawkes process, Survival analysis, Poisson process, Dirichlet process, Low-rank models, Social network analysis, Network structure inference, Information diffusion, Influence estimation, Influence maximization, Submodular maximization, Document clustering, Recommender systems

# CHAPTER I

# INTRODUCTION

Systems in physical nature and human society constantly produce ***asynchronous***, ***interdependent***, and ***high-dimensional*** event data. In social networks, we share our daily experience and respond to other people's information in the form of posts and status updates. In malware monitoring systems, volunteer machines report potentials malicious files to the cloud servers. Although the attacks are reported from different machines at different time, they may be clustered and coordinated, providing a glimpse into the spread of cyber threats across the Internet. In wildlife conservation, we collect observations about the presence of animals, such as birds, migrating across a wide range of geographic locations in certain time periods. In our routine daily lives, people consume personalized news updates, listen to preferred albums, travel to different destinations, and develop a variety of diseases all at different time and space.

Although all these event data arise from a diverse range of domains, they could share common evolutionary dynamics deeply covered by the apparent randomness they appear to be. Due to the fast development of modern communications, sensor networks, mobile devices, and wearable gadgets, the availability and sheer volume of the event data provide unique opportunities for an interdisciplinary research to tackle scientific challenges in innovative ways. Centered around accurately predicting '*who will do what by when and where*', the focus of the dissertation is on developing a novel, efficient, robust and principled machine learning framework for asynchronous and interdependent data which are feature-rich and high-dimensional in big data settings.

## 1.1 Research Problems

The main interest of our research lies in understanding and modeling the asynchronous high-dimensional event data. We investigate the following fundamental research problems and search for an answer to each of them throughout the thesis.

**Continuous-time Information Diffusion.** Networks have been powerful abstractions for modeling a variety of natural and artificial systems that consist of a large collection of interacting and interconnected entities. Due to the recent increasing availability of large-scale networks, network modeling and analysis have been extensively applied to study the spreading and diffusion of information, ideas, and even virus in social and information networks (see *e.g.*, [167, 86, 166]). However, the process of influence and diffusion often occurs in a hidden network that might not be easily observed and identified directly. For instance, when a disease spreads among people, epidemiologists can know only when a person gets sick, but they can hardly ever know where and from whom he (she) gets infected. As a result, we could observe only the time stamp when a piece of information has been received by a particular entity, but the exact path of diffusion is missing. So the first question we ask is:

***Can we uncover the latent dependency structure among entities only based on the observed propagation traces of information (in the form of meme, virus, product)?***

This question is challenging in that many possible latent network structures may exist to explain the observed information traces. However, it is also of extraordinary importance in that its solution is fundamental to better understand the mechanisms governing the diffusion processes of the information among interacting entities. Thus, we devote Chapter 3 to the solution of this question with innovative models and comprehensive evaluations.

**Context-aware Recommendation.** During a particular process for a piece of information to spread through the network, a node can be either infected or survival during the process before a given time $T$, so a single infection event can be used to denote the respective infection moment. In many other real-world systems, however, it is also common that recurrent events occur to the same entity at different time. For instance, users might listen to the same album repeatedly. We visit the same on-line store to buy different products. Based on these recurrent visiting patterns between users and items, we would like to ask:

***Can we recommend the most desirable item to a user at the right moment? Can we predict when a user might come back to existing services in the near future?***

These two questions are related but have different focuses on users' need and behaviors. Successful predictions of the desirable items and the returning time not only allows a service provider to keep track of the evolving user preferences, but also helps them to improve the marketing strategies in order to boost the revenue (e.g. via more accurate user-targeting for advertisement clicks). Chapter 5 discusses these questions and our solutions in detail.

**Temporal Streaming Data Analysis.** Recurrent events not only involve temporal information, but may also include *content-specific features*. For instance, clusters in document streams, such as online news articles, can be induced by the temporal dynamics of their arriving patterns, as well as by their textual contents. As a consequence, we are interested in asking:

***Can we leverage both sources of information to obtain a better clustering of the events, and distill information that is not possible to extract by using either one of the information only?***

In this way, for typical news-feed providers, as each article arrives sequentially,

we are able to extract meaningful descriptions about particular stories (collection of articles about with a central topic) and predict their evolving trends in the future. Compared to the previous problems presented above, this question has its unique challenge in that as streaming events come and go, the number of clusters (or dimensions) can be infinite, and we cannot observe all the events in advance. To address these issues, we discuss our innovative solutions in Chapter 8.

**General Representation of the Influence from Past Recurrent Events.** In context-rich settings, recurrent events most often can bring additional information about their *types*. For instance, people might visit different places at different time of a day. Professional trading systems buy and sell a bunch of stocks within short-time frames. Patients go to see the doctor with a time-stamped sequence of diagnoses of the concerned diseases. In each case, the additional information from the place marker (gas station, airport, etc.), the trading action (buying or selling), and the disease categories constitutes the respective *type* of an event in addition to time. Because this type of extra information can be useful for providing informative clues for accurate predictions of future events, we thus focus on further investigating the following questions:

***Based on the sequence of past events, can we predict what type of event will happen next by when?***

Effective solutions to this question can have significant practical values. For instance, with respect to modern health-care, patients may have several diseases that have complicated dependencies on each other. Accurately estimating when a probable disease might occur can effectively help to take proactive steps to reduce the potential risks in the future. We devote Chapter 7 to the answer of this question.

**Time-sensitive Decision Making.** Successful learning of the accurate predictive models from above enables us to have a better understanding of the underlying dynamics that govern the randomness of the data. Based on these extracted knowledge,

***Can we further make use of the learned dynamics to facilitate time-sensitive decision making processes?***

For instance, can we seed a few websites such that the information can spread as fast as possible within a given time? can we shape the temporal activities of users in a social network? The first problem is a central task in viral marketing, which is often referred to as *influence maximization* where we care about the future adoptions of some one-time actions (buying a product, adopting an idea, retweeting a post, etc.). In contrast, the second problem, referred to as *activity shaping*, is about boosting the intensity of online users' activities (postings, re-tweetings, comments, etc.) which can be recurrent over time. We provide comprehensive solutions and evaluations to these problems in Chapter 4 and Chapter 6.

## 1.2 Contributions and Organizations

To effectively understand the complex dynamics of the event data, throughout this dissertation, we mainly follow three steps:

- **Accurate Modeling:** We propose simple probabilistic models which have intuitive explanations, rigorous mathematical specifications, and accurate predictions about the event data arising from different domains.

- **Efficient Learning:** We develop efficient learning algorithms for fitting the proposed models to large-scale datasets under different structural constraints.

- **Scalable Inference:** Using the extracted knowledge given through the models that explain the data, we design scalable new inference algorithms to make future predictions and controls.

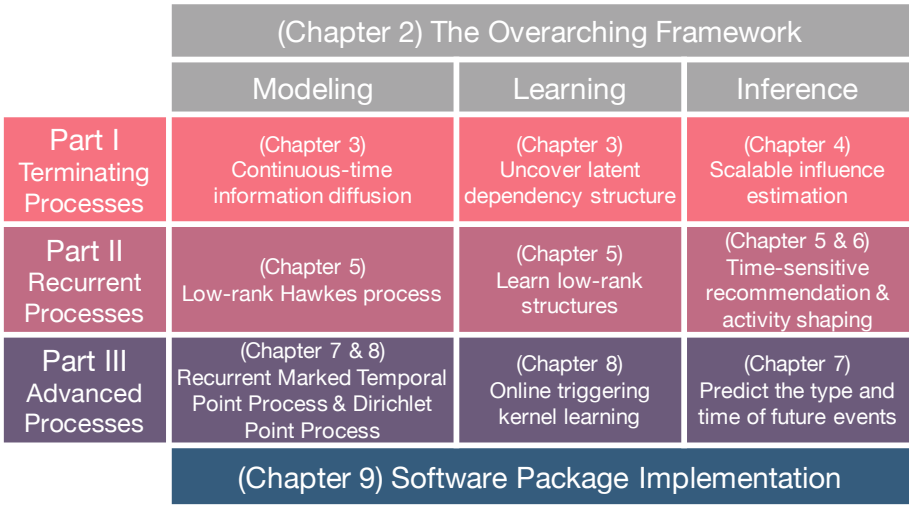



Figure 1.1: Dissertation structure.

The dissertation focuses on three problem domains where the inherent complexity and the effective information of the event data increase one by one. In each problem domain, we present our contributions in the three aspects mentioned above, so the thesis naturally breaks into nine pieces given in Figure 1.1. Specifically, we summarize our contributions as follows:

### Part I: Terminating Processes

- **Accurate Modeling:** we propose novel nonparametric and topic-dependent formulations for multivariate terminating point processes to model the continuous-time information diffusion processes. Our formulations are able to capture general heterogeneous forms of the pairwise transmission functions for the information to spread among entities. Empirical evaluations in both synthetic and real datasets verify that our models have significantly better performance in recovering the latent diffusion networks compared to other state-of-the-arts during the same period.

- **Efficient Learning:** we develop a robust structure learning algorithm via group lasso based on an elegant convex formulation. The algorithm is able

to efficiently uncover sparse heterogeneous interdependent relations specified via vectorized parameters among the dimensions and can execute in parallel across multiple computing nodes.

- **Scalable Inference:** we develop a randomized algorithm for scalable influence estimation in continuous-time diffusion networks. Our algorithm can estimate the influence of every node in a network with $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges to an accuracy of $\epsilon$ using $n = O(1/\epsilon^2)$ randomizations and up to logarithmic factors $O(n|\mathcal{E}| + n|\mathcal{V}|)$ computations. When used as a subroutine in a greedy influence maximization approach, our proposed algorithm is guaranteed to find a set of $C$ nodes with the influence of at least $(1 - 1/e)\,\mathrm{OPT} - 2C\epsilon$, where OPT is the optimal value. Another novelty of our work is that we are the first to evaluate the solution quality given by the algorithms against real independent testing diffusion data, which further verifies that the produced solutions can indeed generate larger influence compared with other state-of-the-arts.

### Part II: Recurrent Processes

- **Accurate Modeling:** we present a new low-rank Hawkes process for capturing the recurrent actions between users and items to make time-sensitive recommendations and returning time predictions. Experimental evaluations on synthetic and real datasets verify that the new model can achieve superb predictive performance compared to other state-of-the-arts.

- **Efficient Learning:** we develop a new optimization algorithm to learn the low rank Hawkes process model efficiently. Our algorithm blends proximal gradient and conditional gradient methods, and achieves the optimal $O(1/t)$ convergence rate. As further demonstrated by our numerical experiments, the algorithm is able to scale up to millions of user-item pairs and hundreds of millions of temporal events.

- **Scalable Inference:** based on the models for recurrent user activities, we propose the first user-activity shaping problem for multivariate Hawkes processes. We derive a novel predictive formula for the overall network activity given the intensity of exogenous events of each individual user. We also propose a convex optimization framework to address a diverse range of activity shaping problems given certain budget constraints. Compared to previous methods for influence maximization, our framework can provide more fine-grained control of network activity, not only steering the network to a desired steady-state activity level but also do so in a time-sensitive fashion.

### Part III: Advanced Processes

- **Accurate Modeling:** we propose the Recurrent Temporal Point Process to jointly model the time and the marker information of events by formulating a general nonlinear dependency over the history. The model establishes a novel connection between recurrent neural networks and point processes, which has implications beyond temporal settings by incorporating more rich contextual information and features. Furthermore, we design the Dirichlet Point Process by building a previously unexplored connection between Bayesian Nonparametrics and Temporal Point Processes, which allows the number of dimensions to grow in order to accommodate the increasing complexity of online streaming data, while at the same time learns the ever changing latent temporal dynamics.

- **Efficient Learning:** we develop an efficient stochastic gradient algorithm for learning the Recurrent Temporal Point Process which can readily scale up to millions of events. Comprehensive experiments on both synthetic and real-world datasets across a diverse range of domains to show that we have consistently better predictive performance for both the event type and timing compared to alternative competitors.

- **Scalable Inference:** we propose an efficient online inference algorithm which can scale up to millions of news articles with near constant processing time per document and moderate memory consumption. We conduct large-scale experiments on both synthetic and real-world datasets to show that Dirichlet Point processes can recover meaningful topics and temporal dynamics, leading to better predictive performance.

The remainder of the thesis is organized as follows. Chapter 2 first presents the overarching framework and necessary concepts needed throughout the thesis. Then, each following chapter focuses on one aspect of the framework. Specifically, we investigate continuous-time information diffusions in Chapter 3 and 4. Chapter 3 is devoted to recovering the latent diffusion network structure and pairwise diffusion dynamics, based on which Chapter 4 further tackles the scalable time-sensitive influence estimation and maximization problems. Next, we study recurrent user activities in Chapter 5 and 6. Chapter 5 focuses on modeling the recurrent interactions between users and items, where we describe the Low-rank Hawkes Process for time-sensitive recommendations and present a new convex optimization algorithm for fitting the model. Chapter 6 follows to model the recurrent activities among users and presents efficient algorithms to shape user activities. Chapter 7 presents the recurrent marked temporal point process for learning a general representation of the dependency over the past events to predict both the timing and the type of future events. Chapter 8 introduces the Dirichlet point process and an efficient online inference algorithm to combine the modeling of the temporal information with additional textual data. We introduce our open source package `PtPack` in Chapter 9, which efficiently implements the proposed framework of the thesis. Finally, we conclude with future research directions in Chapter 10.

# CHAPTER II

# THE OVERARCHING FRAMEWORK

In this chapter, we first introduce the motivation and overview for event data analysis. Then, we explain our proposed framework via multivariate point processes, which explicitly captures the rate of event occurrence and their interdependency as a function of the historical data.

## 2.1 Motivation

Natural and artificial systems in the physical world and human society often produce sheer volumes of events in various forms induced from the complex collective interactions among the involved entities. We take concrete examples from the following diverse domains to demonstrate the pervasiveness of these event data.

**Online systems.** Billions of users (each considered as a dimension) in social networks generate myriads of posts (events) daily. Moreover, these events often exhibit sophisticated dependencies over each other. One popular post (event) of a user can trigger a sequence of following events (retweets, comments) from the same or other users, forming an information cascade in the system. With modern internet streaming services, people consume news-feeds, listen to preferred tracks, watch entertaining videos at different time and places. These recurrent activities occurring randomly could reflect the evolving interests of users under different contexts.

**Cyber security.** In malware monitoring systems, such as Symantec worldwide cyber intelligence network environment[1], 1.6 million hosts (each considered as a dimension) worldwide report 22 million malicious files (events) in a single month [127]. Although the attacks were reported from geographical spread of different machines at different time, they may be clustered and coordinated to form propagation patterns, providing a glimpse into the spread of cyber threats across the web.

**Wildlife conservation.** To study strategies of wildlife conservations, researchers collect observations about the presence (events) of certain specifies, such as the birds[2], to provide rich data for basic information about their abundance and distribution at a variety of locations (each considered as a dimension) across a wide temporal scale. These observations collectively could reflect the migration patterns of the animals. If we can accurately uncover their migration paths, we can design better planning to connect core habitat areas in a way that is robust to unexpected natural or anthropological disturbances under limited resources [98].

**Modern health-care.** Critical care medicine costs have been increasing annually over the past decade. Patients may have several diseases (each considered as a dimension) that have complicated dependencies on each other. At each hospital visiting, the occurrence of one disease (event) might trigger the progression of other diseases. Due to the increasing adoption of the electronic health records (EHR), earlier detection of disease progression can effectively reduce the potential risk of future critical care for the patients.

**Financial domain.** In stock market, millions of transactions (events) occur to each stock (considered as a dimension) per day. Normally, the rise and fall of one

[1] http://www.symantec.com/about/profile/universityresearch/sharing.jsp
[2] http://ebird.org/content/ebird/about/

stock might trigger a ripple effect on other stocks. Accurate forecasting the market trend can thus be critical for sustainable business success. Similarly, for peer-to-peer (P2P) micro-finance services[3], they provide precious opportunities to help young entrepreneurs. Each transaction (event) provides rich information about what type of loans a lender (dimension) seeks to finance. Identifying the key temporal factors and loan features to provide a reliable and vibrant financial setting is of outstanding importance to the engagement of users and the healthy development of the whole P2P financial community.

In all these examples, both asynchronous event timing information (or temporal dynamics) and observation features carry a great deal of information about the behavior of the involved entities (eg. users, hosts, habitats, diseases, or lenders). Many applications and scientific challenges arising from these domains thus require a representation and analysis of these asynchronously and interdependently generated high dimensional event data. In this type of data, each of the dimensions is observed at different time points, and the feature and timing of each observation can be affected by past observations.

Previously, problems in social networks, malware detection, computational sustainability have been studied predominantly using traditional independently-and-identically distributed (*i.e.*, iid.) models, time-series models, static graph or graph time-series models. However, all these traditional methods cannot be naturally applied to the large volume of high dimensional asynchronous and interdependent event data collected in the problem domains mentioned above. First, this type of event data are very different from the traditional independently and identically distributed (*i.e.*, iid.) data where all dimensions of a data point are collected simultaneously, and the data points are independently sampled from an identical distribution. Therefore,

[3]www.kiva.org

iid models are too restrictive by completely ignoring the temporal dependency. Although the less restrictive time-series models can capture the dependency relation to certain extent, they need to discretize the timeline into windows, and aggregate information within each time window artificially. As a result, important temporal dynamics can be lost due to such coarsening. Furthermore, these event data are also very different from time-series data where the dimensions of a data point are still measured simultaneously though the data points can be dependent and time intervals between two adjacent data collections can vary. Static graphical models explicitly describe the relations between interacting entities, but it is not easy to incorporate the event timing information in these models. Finally, graph time-series models can capture the evolution and time-varying nature of the relations to some extent, but again arbitrarily dividing the timing information into epochs makes these methods incapable of answering many questions related to temporal dynamics.

With the advent of a sheer volume of asynchronous and interdependent data routinely generated across a variety of domains from social networks to computational sustainability to medical informatics, novel methodologies need to be designed to tackle the new challenges arising from these applications.

## 2.2 Literature Survey

Over the years, a large collection of sophisticated predictive models have been developed and extensively discussed in various literatures [70, 144, 145, 19, 92, 119] of the machine learning community from logistic regression to support vector machines to feedforward networks based on the primary assumption that data samples are independently and identically distributed. This assumption of independence makes many learning tasks more easier by expressing the joint likelihood of observing all the data samples as a product of individual likelihood evaluated at each single data point. However, despite the usefulness of the *i.i.d.* assumption, for many practical

applications, this assumption will be a bad idea in that it precludes the modeling of the dependencies. To alleviate this issue, time-series models, static graphical models, and graph time-series have been proposed seeking to capture the temporal and the structural dependencies. Unfortunately, almost all of the methods fail to explicitly address applications with asynchronously generated dependent data.

**Time-series models.** Conventional time-series models, like Markov Chain [19], Varying order Markov model [18], Hidden Markov Model [16], Kalman Filters [82], Vector auto-regressive models [65] are extensively used for modeling sequential data. However, one major limit of these models is that the time stamp is implicitly treated as the index of discretized unit steps. Because the system evolves in a synchronized step-by-step fashion, these models fail to capture the asynchronous characteristics of the event data. A proper step size is a free parameter which is normally hard to tune practically for that we need to aggregate multiple data samples within the same unit step and consider the case when there is no data point in a unit step at all. Although the Semi-Markov model [81] seeks to alleviate this issue by explicitly modeling the transition time between two states as continuous exponential random variables, such fixed parametric form of the exponential distribution and the limited short-term temporal dependency over the history have limited their usefulness in practice. Finally, most applications of the conventional time-series models only involve one or low dimensional data, like speech signals, this makes them insufficient to capture the complex interdependency structure among different components for the multi-dimensional data.

**Dependency Structure Learning.** Although it is relatively easy to collect sets of observed event data due to modern communication techniques, the strength of interplays and the structure of interdependency between different dimensions might

be partially observed or even fully hidden. For instance, we might record when people go to see the doctor when they are sick, but we may hardly observe who has infected whom. As a result, it is still challenging to estimate and uncover the latent dynamics governing these seemingly random occurrences of the events to each dimension. Traditionally, probabilistic graphical models [52, 53] are widely used for inferring the static structure among different variables. While there is a rich and growing literature on learning time-invariant networks from *i.i.d.* observations and designing efficient algorithm for the estimation procedure, *e.g.*, [136], much less has been done toward modeling the dynamics of the networked entities. An exception is the continuous-time Bayesian network [123] which is essentially a structured Markov process. However, it is still restricted in the sense that each dimension is a homogeneous Markov process which cannot capture the long-term temporal dependency over the past events of its own and other dimensions. Furthermore, being similar to the one-dimensional continuous-time Markov chain, the time duration between two successive events is limited to the exponential distribution, which cannot model the general heterogeneity of the temporal dynamics. Recently, a sequence of work [149, 150, 5, 91, 90, 74] seek to learn the time-varying dependency structure from time-series data. However, the time stamps again are treated as discrete indices, and thus they still cannot capture the general temporal dynamics associated with each edge which collectively govern the evolution of the overall network structure by time.

**Modeling Social Interactions.** In the community of social network analysis, modeling rich social interactions can be traced back to the link prediction problem first studied by [104] where different score functions were applied based on the structural properties of the network to predict the existence of a link based on thresholding values. In [2], Adar et al. formulate the problem as a binary classification task with rich contextual and structural features extracted from the network to predict whether a

single link might exist between a given pair of nodes. Although many rich interaction features are used, each link is predicted locally and independently. Thus, static link prediction [108, 69] cannot capture the interdependency among different edges from a global optimization perspective. Recently, modeling dynamic interactions of users in social networks has been attracting increasing interests. A collection of dynamic models about network formation and evolution is reviewed in [54, 76]. Backstrom et al. [9] investigate the mechanism governing group formation and growth based on user interactions. Wouter de Nooy in [36] studies the timing of relational events (edges appear, change or disappear) on a set of longitudinal social networks. Matsubara et al. [114, 116, 113] model macroscopic online user activities and event patterns based on tensor decomposition and ecological population formulations. Matsubara et al. [115] also develop generative models to capture the rise and fall patterns of information cascades. However, most of these models still treat the timing information as discrete indices. Snijders [148] used a continuous-time Markov process with exponential random graphs to model friendship temporal dynamics. Brandes *et al.* explicitly model the probability density for the observed event sequences over a given network by incorporating social science theory for longitudinal networks. However, the restricted model flexibility of these work makes them fail to describe the interdependency between history events of the same dimension or between those from different dimensions, which is one of the focuses in this dissertation on the high-dimensional data in the big-data settings.

**Point Process.** Multivariate point processes are mathematical tools for modeling multidimensional event data [1]. However, only very recently machine learning community are starting to use it for modeling practical problems. For instance, these mathematical tools has been applied to analyze activities [110, 109] and criminal

Table 2.1: Table of symbols

| Symbol | Description |
|---|---|
| $t_i$ | Occurrence time for the $i$-th event |
| $t_i^d$ | Occurrence time for the $i$-th event on dimension $d$ |
| $\mathcal{H}_t$ | List of event time $\{t_1, t_2, \ldots, t_n\}$ up to but not including time $t$ |
| $f^*(t), f(t\|\mathcal{H}_t)$ | Conditional density function for the next event to occur at time $t$ |
| $F^*(t), F(t\|\mathcal{H}_t)$ | Cumulative density function for the next event to occur before $t$ |
| $S^*(t)$ | Survival probability up to time $t$ |
| $\lambda^*(t)$ | Conditional intensity function |
| $\lambda_d^*(t)$ | Conditional intensity function on dimension $d$ |
| $p_d^*(x)$ | Conditional density for the feature $x$ |
| $D$ | Number of dimensions |
| $n_d$ | Number of events on dimension $d$ |
| $T$ | Observation window |
| $\text{Exp}(x)$ | Exponential distribution |
| $\text{Uniform}(x)$ | Uniform distribution |

gang rivalries [117, 118, 152, 48, 146, 131], to understand interactions in communication networks [160, 23, 161, 64, 112, 161]. However, the limitation of these previous works is that they do not model the interdependency between different users and can only be applied to relatively small dimensions. In contrast, we seek to develop a more general framework which will address a broader range of problems arising from asynchronously generated interdependent high-dimensional event data. Furthermore, in the case of high dimensional multivariate point processes, the dependency structures parameters between the dimensions are often unknown. It is an interesting and challenging question whether we can uncover these dependency structures based on the time stamps and features of the events. This problem has been addressed only by a paucity of recent studies in the literature [58, 56], and there is a whole range of other inference tasks which have not been addressed by the previous work. To facilitate the description of the framework, the accompanying table provides a list of symbols and their definitions.

## 2.3 A Unified Probabilistic Framework

We propose a framework based on multivariate point processes. This framework *explicitly* models the rate of event occurrence as a direct function of the timings and possible features of past events, *effectively* uncovers the latent interdependency structure among the dimensions with the respective heterogeneous temporal dynamics, and *efficiently* carries out large-scale inference tasks based on the learned models. It consists of three methodological parts: (I) accurate and robust ***modeling*** of event dynamics; (II) efficient ***learning*** of the predictive models over large-scale datasets; and (III) scalable ***inference*** based on the learned models for time-sensitive decision makings.

### 2.3.1 Modeling

Formally, a temporal point process is a random process whose realization consists of a list of discrete events localized in time, $\{t_i\}$ with $t_i \in \mathbb{R}^+$ and $i \in \mathbb{Z}^+$. Let the history $\mathcal{H}_t$ be the list of event time $\{t_1, t_2, \ldots, t_n\}$ up to but not including time $t$. The lengths of the time intervals between neighboring successive events are referred to as the inter-event times. Given the history of past events, we can explicitly specify the conditional density function that the next event will happen at time $t$ as $f^*(t) = f(t|\mathcal{H}_t)$ where $f^*(t)$ emphasizes that this density is conditional on the history. By applying the chaining rule, we can explicitly derive the joint density of observing a sequence $\mathcal{S}^i$ as

$$f\left(\{t_i\}_{i=1}^n\right) = \prod_i f(t_i| \ldots, t_{i-2}, t_{i-1}) = \prod_i f^*(t_i) \tag{2.1}$$

A temporal point process can be equivalently represented as a counting process, $N(t)$, which records the number of events before time $t$. Let the history $\mathcal{H}_t$ be the list of event time $\{t_1, t_2, \ldots, t_n\}$ up to but not including time $t$. Then in a small time window $dt$ between $[0, t)$, the number of observed event is

$$dN(t) = \sum_{t_i \in \mathcal{T}} \delta(t - t_i)\, dt, \tag{2.2}$$

and hence $N(t) = \int_0^t dN(s)$, where $\delta(t)$ is a Dirac delta function. It is often assumed that only one event can happen in a small window of size $dt$, and hence $dN(t) \in \{0, 1\}$.

An important way to characterize temporal point processes is via the conditional intensity function — the stochastic model for the next event time given all previous events. Within a small window $[t, t+dt)$, $\lambda^*(t)dt$ is the probability for the occurrence of a new event given the history $\mathcal{H}_t$:

$$\lambda^*(t)dt = \mathbb{P}\left(\{\text{event in } [t, t+dt)\} \,|\mathcal{H}_t\right). \tag{2.3}$$

Again, the $*$ notation reminds us that the function depends on the history. Intuitively, the conditional intensity function denotes the *risk* or *hazard rate* that an event can happen at time $t$. For instance, if an event represents being infected by a flu, then the intensity value at any given time $t$ of this event is normally higher on average for the elders than the young adults even though both groups are in healthy status, so the intensity function $\lambda^*(t)$ does not indicate an event will occur at time $t$, but rather it quantifies the risk at which an event can happen. Based on this intuition, we can have the following basic derivations:

$$\begin{aligned}
\lambda^*(t)dt &= \mathbb{P}\left(\{\text{event in } [t, t+dt)\} \,|\mathcal{H}_t\right) \\
&= \mathbb{P}\left(\{\text{event in } [t, t+dt)\} \,|\, \{\text{no event up to } t\}, \mathcal{H}_t\right) \\
&= \frac{\mathbb{P}\left(\{\text{event in } [t, t+dt)\} \text{ and } \{\text{no event up to } t\} \,|\mathcal{H}_t\right)}{\mathbb{P}\left(\{\text{no event up to } t\} \,|\mathcal{H}_t\right)} \\
&= \frac{f^*(t)dt}{1 - F^*(t)} = \frac{f^*(t)dt}{S^*(t)},
\end{aligned} \tag{2.4}$$

where $F^*(t)$ is the cumulative probability that a new event will happen before time $t$ since the last event time $t_n$, and $S^*(t) = 1 - F^*(t)$ is the respective probability that no new event has ever happened up to time $t$ since $t_n$, which is also known as the survival probability. In addition, the conditional intensity function $\lambda^*(t)$ is also related to the survival probability $S^(t)$ via the differential equation

$$\lambda^*(t) = \frac{-d \log S^*(t)}{dt}. \tag{2.5}$$

Solving the differential equation with the boundary condition $S^*(t_n) = 1$, we arrive at the following basic relation:

$$S^*(t) = \exp\left(-\int_{t_n}^{t} \lambda^*(\tau) d\tau\right), \tag{2.6}$$

where $t_n$ is the last event in $\mathcal{H}_t$. As a consequence, the conditional density function can be alternatively specified by

$$f^*(t) = \lambda^*(t) \exp\left(-\int_{t_n}^{t} \lambda^*(\tau) d\tau\right). \tag{2.7}$$

Particular functional forms of the intensity $\lambda^*(t)$ are often designed to capture the phenomena of interests [1]. For example, the homogeneous Poisson process is the simplest point process. The inter-event times are independent and identically distributed random variables conforming to an exponential distribution. The conditional intensity function is assumed to be independent of the history $\mathcal{H}_t$ and constant over time, *i.e.*, $\lambda^*(t) = \lambda_0 \geqslant 0$.

For most real problems in practice, the event data is normally high-dimensional. For example, in social networks, each user is regarded as a dimension, and we have billions of users generating tweets everyday. In medical informatics, each disease can be a dimension, and we have thousands of different types of diseases. Furthermore, we might also have additional contextual feature information associated with each event, such as the topic of each tweet, the physical condition of each patient, etc. These additional contextual features might also influence the underlying evolutionary process of the events. Therefore, to model these feature-rich high-dimensional data, we can generalize existing theories to the multi-dimensional case.

Specifically, let $d \in \{1, 2, \ldots, D\}$ be the dimension of the point process. We will treat the $i$-th event as a triplet of the form $(t_i, d_i, x_i)$ $(i = 1, \ldots, n)$ where $t_i \in \mathbb{R}^+$, $d_i \in \{1, \ldots, D\}$ and $x_i \in \mathcal{X}$. This means that we order events according to their timing information, and the $i$-th event occurs at $d_i$-th dimension with additional features vector $x_i$ from some feature space $\mathcal{X}$. Such representation explicitly represents timing

as a source of information and emphasizes the asynchronicity and the feature-richness of the events.

In general, the conditional intensity $\lambda_d(t|\mathcal{H}_t)$ can depend on historical events from both dimension $d$ and other dimensions. Denote the history at time $t$ as a collection of event before time $t$, ie., $\mathcal{H}_t = \{(t_1, d_1, x_1), \ldots, (t_i, d_i, x_i)|t_1 < \ldots < t_i < \ldots < t\}$. Then $\lambda_d^*(t) = \lambda_d(t|\mathcal{H}_t)$ is again a general nonnegative function of all events in $\mathcal{H}_t$. For each dimension $d$, the structure of the conditional intensity function explicitly captures the general interdependent relations among different dimensions. However, because these interdependent relations are often partially observed or even completely hidden, one focus of this framework is to incorporate different parametric and nonparametric formulations to capture different phenomena of interests and unveil such latent interdependent relations and their respective strengths from the generated event data in a unified way.

Besides, in addition to the conditional intensity function, we can also model generative processes for the feature by formulating its conditional density function $p_d^*(x) = p_d(x|t, \mathcal{H}_t)$. Depending on the data we want to capture and the type of features, we can choose very different generative models for $p_d^*(x)$. For the simplest case, suppose the feature value is independent of the history $\mathcal{H}_t$ and $t$, then for discrete features, we can use the Multinomial distribution, while for continuous features, we can use the Gaussian distribution.

In order to capture the different characteristics of the asynchronous high-dimensional event data arising from a wide range of domains, the proposed framework has comprised the following innovative models:

**Multivariate Terminating Process.** In many cases, people need to analyze the time until a single event occurs to each entity. Some examples include the timing of adopting an idea, purchasing a product, catching a virus, experiencing a failure,

*etc.* Moreover, in a networked setting, the occurrence of one event to an entity can be triggered by the past occurrences of events to the other entities due to their local interactions, which in turn often results in a global cascade or diffusion over the network along time. Therefore, we are interested in asking: how fast can a piece of information propagate in a network? can the spreading be predicted? The multivariate terminating processes are thus designed to address these questions by accurately capturing a general continuous-time information diffusion process over a network of interacting entities.

**Low-rank Hawkes Process.** Most often, an event in question may occur more than one time for an entity. Examples of such recurrent events can be repeated user activities in online social networks, frequent transactions in trading systems, progressive tumors in cancer studies, *etc.* Furthermore, these recurrent events can also be correlated with each other. In social network analysis, users of the same community tend to behave similarly. In cancer studies, patients with similar phenotypes may go through similar progressive stages of a particular disease. In all these cases, although the dimension of the event data can large (billions of online users, millions of patients), such correlations may allow us to capture their interactions in a lower-dimensional space. To this end, the proposed low-rank Hawkes process is able to explicitly model the low intrinsic dimensionality of the interdependency structure among the different types of entities.

**Recurrent Marked Temporal Point Process.** Sometimes, besides time, extra information, known as the *markers*, is also available with each event. Examples include the magnitude of an earthquake, the passenger pick-up place of a taxi, the buying or selling action conducted on a stock, *etc.* Are these additional information helpful for predicting the occurrences of future events? The recurrent marked temporal point process is designed to effectively capture and represent the marker

information to help us to better understand the latent evolutionary dynamics of a temporal sequence and to further improve the performance of predicting new events in the future.

**Dirichlet Point Process.** Each marker associated with a temporal event often carries limited information about its type and characteristics. For many user-generated data, like tweets, documents, news articles, etc., rich content information is also an important source for us to organize and extract meaningful insights about the subjects being studied. How can we distill more knowledge from the data by jointly considering both the temporal and textual information than by using either one of them alone? In this regard, the Dirichlet point process is proposed to extend the flexibility of our framework through the incorporation of other type of data into the evolutionary process of the temporal events.

### 2.3.2 Learning

Based on the elegant connection between the conditional intensity function and the density function in (2.7), the conditional density for an event $(t_i, d_i, x_i)$ given the history $\mathcal{H}(t)$ can be expressed as

$$f_{d_i}(x_i, t_i|\mathcal{H}_{t_i}) = p^*_{d_i}(x_i)\ \lambda^*_{d_i}(t_i)\ \exp\left(-\int_{t_{i-1}}^{t_i} \lambda^*_{d_i}(s)\ ds\right). \tag{2.8}$$

By the chain rule, the joint likelihood of observing a single sequence $\{(t_i, d_i = d, x_i)\}_{i=1}^{n_d}$ on the $d$-th dimension where $n_d$ is the total number of events on dimension $d$ can be given by

$$L_d\left(\{(t_i, d_i = d, x_i)\}_{i=1}^{n_d}\right) = \prod_{i=1}^{n_d} f_d(x_i, t_i|\mathcal{H}_{t_i}) \cdot (1 - F^*_d(T)), \tag{2.9}$$

where the last term is the probability that there is no event between $t_n$ and $T$ on the $d$-th dimension. Taking the log, we can get the log-likelihood by:

$$\begin{aligned}
&\ell_d\left(\{(t_i, d_i = d, x_i)\}_{i=1}^{n_d}\right) = \log\left(\left(\prod_{i=1}^{n_d} f_d(x_i, t_i|\mathcal{H}_{t_i})\right) S_d^*(T)\right) \\
&= \log\left(\left(\prod_{i=1}^{n_d} p_d^*(x_i)\ \lambda_d^*(t_i) \exp\left(-\int_{t_{i-1}}^{t_i} \lambda_d^*(\tau) d\tau\right)\right) \exp\left(-\int_{t_{n_d}}^{T} \lambda_d^*(\tau) d\tau\right)\right) \\
&= \sum_{i=1}^{n_d} \log p_d^*(x_i)\ \lambda_d^*(t_i) - \int_0^T \lambda_d^*(\tau) d\tau. \qquad (2.10)
\end{aligned}$$

Then, the overall log-likelihood of observing the events on all dimensions $\{(t_i, d_i, x_i)\}_{i=1}^{n}$ can be easily decomposed into a summation of the individual log-likelihood $\ell_d$ of the sequence associated with each dimension $d$.

$$\ell\left(\left\{(t_i^d, x_i^d)\right\}_{i=1\ldots n_d}^{d=1\ldots D}\right) = \sum_{d=1}^{D} \ell_d\left(\{(t_i, d_i = d, x_i)\}_{i=1}^{n_d}\right). \qquad (2.11)$$

Depending on the different formulations of $\lambda_d^*(t)$ and $p_d^*(x)$, we can maximize the joint log-likelihood in (2.11) subject to the specific structural constraints of different problems in order to fit the process to the observed point patterns. Unfortunately, only for a few simple cases, the optimization of 2.11 can have the closed form solution. For example, for the homogeneous Poisson process, where $\lambda_d^*(t) = \alpha_d$ is a single constant, the joint log-likelihood of (2.11) without considering the features simply reduces to

$$\ell\left(\left\{t_i^d\right\}_{i=1\ldots n_d}^{d=1\ldots D}\right) = \sum_{d=1}^{D} \left(n_d \log \alpha_d - T\alpha_d\right), \qquad (2.12)$$

which is concave with respect to each parameter $\alpha_d$. Thus, we can take the derivative and set it to be zero to get the global optimal estimation as

$$\hat{\alpha_d} = \frac{n_d}{T}, \qquad (2.13)$$

which is the average number of events occurring per unit time on each dimension. In most cases, maximizing (2.11) does not have closed form solutions, so we need

general numerical optimization techniques to estimate the parameters. By explicitly exploiting the specific properties of the optimization problem in 2.11 induced from particular parameterizations of the conditional intensity functions under various structural constraints, we have developed the following algorithmic components in the framework:

**Sparse Network Structure Learning.** In most cases, the interdependent structure among the dimensions is not fully observed. Moreover, the interactions between pairs of dimensions can be wildly diverse, so the respective pairwise temporal dynamics can be heterogeneous, which might not be well captured by a single family of parametric models. In order to capture such heterogeneity of pairwise interactions, we propose nonparametric formulations, where the parameter space has a inherent group structure, and develop efficient proximal gradient methods to uncover the sparse and heterogeneous interdependency between the dimensions.

**Efficient Nonnegative Matrix Rank Minimization.** In addition to the sparsity and the heterogeneity, in specific problem domains, such as the recommender systems, the interdependency among the dimensions, which can be conveniently represented as a matrix, also has an essential low-rank structure. However, directly maximizing (2.11) under the low-rank constraint and the non-negativity constraint required by the intensity function is often challenging. To tackle this problem, we develop a new optimization algorithm which inherits from the advantages of both proximal and conditional gradient methods to efficiently learn the model from hundreds of millions of events.

**Efficient Online Bayesian Updating.** In the online stream setting where the temporal event happens one after another, efficient updating of the model parameters is critical to maintain a sustainable performance of the online system. We propose

an efficient online Bayesian updating algorithm, of which the average time cost for processing each data point can maintain roughly constant after keeping running for a long time period.

### 2.3.3 Inference

Given that we have developed sophisticated predictive models and accurately fit them to the data from the previous two components of our framework, how can we utilize the extracted insights from the models to make predictions of the future? The third component of the framework is thus devoted to a variety of inference tasks across different domains.

For classic event data analysis, simulation of new events is the most common task based on a given point process. Depending on the formulation of the intensity function $\lambda^*(t)$, the conditional density $f^*(t)$ for the next event may or may not have a close form. Given the history of past events $\mathcal{H}_t$, we can use the expectation taken with respect to $f^*(t)$ as our best estimation for the next event time. When $f^*(t)$ does not have a close form solution, we can instead generate sufficient number of simulations for the next event, and take the average as our prediction. Similarly, many other quantities that depend on the conditional intensity function might be difficult to compute directly as well. For instance, the average number of events in a given interval might not have a close form solution in that the integral of $\lambda^*(t)$ can be quite complicated. As a consequence, we can again turn to the approximate solution by using a number of simulated samples. Finally, given a specified form for the conditional intensity $\lambda^*(t)$, simulation is also a direct way to visualize the point patterns, providing valuable information about the respective point process.

In addition to the basic simulation of temporal events, our framework supports many more sophisticated inference tasks as follows:

**Influence Estimation.** If a piece of information is released from a media site, can we predict how many websites will mention it in a month? Essentially, the solution to this kind of problem requires us to estimate the influence of a given node (or dimension) in a networked setting within a predefined time window. By exploiting the learned multivariate terminating point process, we propose a randomized algorithm for efficient influence estimation that can scale linearly up to networks of millions of nodes, which is the key for many applications in viral marketing.

**Time-Sensitive Recommendation.** In personal assistant systems, can we make recommendations specific to the temporal/spatial contexts of users? In modern health-care, can we provide accurate predictions of the timing when a particular disease may happen to a given patient? Based on the multivariate low-rank Hawkes process, we design new time-sensitive recommendation algorithms in the framework to make context-aware predictions to the future.

**User Activity Estimation.** Online users' activities (events) can be triggered either by *endogeneous* events, where users simply respond to the actions of their neighbors within a network, or by *exogenous* events, where they take actions due to drives external to the system. Since active engagement of users is crucial for the healthy growth of business for modern web companies, by utilizing the multivariate Hawkes process and its connection to the Branching process, we propose novel formulations for calculating the expected engagement level of users at any given time in large networks.

**Online Inference for Document Cluster.** In online document streams, accurately gathering related news articles centered around a particular story based both textual and temporal similarity can effectively help people to extract meaningful knowledge from the sheer amount of information. Based on the proposed Dirichlet

point process, we design an efficient online inference algorithm using particle filters which can scale up to millions of news articles with near constant processing time per document and moderate memory consumption.

Scalable algorithms for solving these fundamental inference problems make it possible for us to address many challenging time-sensitive decision making problems efficiently. For instance, how can we select a small set of earlier adopters to maximize the spreading of a new product within a short-time window? how can we promote the interests of online users to sustain the growth of a community? We will discuss novel solutions to these inference and optimization problems in the following chapters.

## 2.4 Summary

Modern applications and scientific problems arising from a variety of domains from social network and online media analysis to human activity modeling to medical informatics to financial transactions call for representation and analysis of large-scale asynchronous high-dimensional event data. A basic premise of our research is that asynchronous feature-rich event temporal dynamics carry massive information about the microscopic behaviors of the entities in study, and our primary goal is to be able to address the *who will do what by when and where?* question with exploratory and predictive models. The focus of this dissertation is thus on a unified probabilistic framework based on multivariate point processes for statistical modeling, learning, and inference in large-scale asynchronous event data. The framework consists of three inter-related pillars: I. accurate and robust predictive models; II. efficient learning algorithms; and III. large-scale probabilistic inferences. We then proceed in each chapter of the thesis to demonstrate different capabilities of this framework in solving real-world problems across a wide range of domains.

# PART I TERMINATING PROCESSES

Event time data contain rich information about the underlying generative processes which might be hidden or partially observed. Most often, people need to analyze the time until a specific event occurs. Some concrete examples include the timing of adopting a piece of information, purchasing a product, catching a virus, experiencing a failure, *etc.* In all these cases, the response of interest is the time duration from a predefined time origin to the occurrence of some given event (or terminating-point). Moreover, in a networked setting, the occurrence of one event to an entity can be triggered by the past occurrences of events to the other entities due to their local interactions, which in turn often results in a global cascade or diffusion over the network along time. In the first part of the thesis, we systematically investigate the general temporal dynamics of continuous-time information diffusions (Chapter 3) and seek to design algorithms that can predict and optimize the spreading processes (Chapter 4) in the future.

**Accurate Modeling:** we propose novel nonparametric and topic-dependent formulations for multivariate terminating point processes to model the continuous-time information diffusion processes.

**Efficient Learning:** we also develop a robust structure learning algorithm via group lasso to efficiently uncover sparse heterogeneous interdependent relations among different dimensions.

**Scalable Inference:** based on the learned multivariate terminating process, we develop a randomized algorithm for general influence estimation and maximization in continuous-time diffusion networks.

# CHAPTER III

# CONTINUOUS-TIME INFORMATION DIFFUSION

Diffusions of information, ideas, fashions, behaviors, and diseases over networked entities are universal phenomena in systems of nature and our society. In online social networks, widespread of entertaining videos makes people famous and rich. In local regions, outbreaks of infectious diseases cause people sick and dead. How do information and diseases propagate? How fast the diffusion process can be? How can we estimate and eventually control the spreading trend in the future?

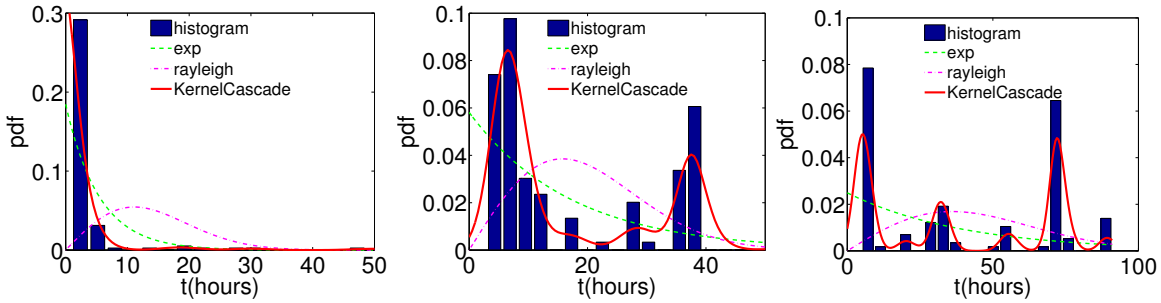
In this chapter, we propose a simple but accurate predictive model to explain the heterogeneity of the temporal dynamics which inherently govern the information diffusion processes. Then, we develop efficient learning algorithms from elegant convex formulations to estimate the model parameters. In both synthetic and real cascade data, we show that our model can better recover the underlying diffusion network and drastically improve the estimation of the transmission functions among networked entities.

## 3.1 Introduction

Networks have been powerful abstractions for modeling a variety of natural and artificial systems that consist of a large collection of interacting entities. Due to the recent increasing availability of large-scale networks, network modeling and analysis have been extensively applied to study the spreading and diffusion of information, ideas, and even virus in social and information networks (see *e.g.*, [167, 86, 166]). However, the process of influence and diffusion often occurs in a hidden network that might not be easily observed and identified directly. For instance, when a disease spreads among people, epidemiologists can know only when a person gets sick, but

they can hardly ever know where and from whom he (she) gets infected. Similarly, when consumers rush to buy some particular products, marketers can know when purchases occurred, but they cannot track in further where the recommendations originally came from [103]. In all such cases, we could observe only the time stamp when a piece of information has been received by a particular entity, but the exact path of diffusion is missing. Therefore, it is an interesting and challenging question whether we can uncover the diffusion paths based just on the time stamps of the events.

There are many recent studies on estimating correlation or causal structures from multivariate time-series data (see *e.g.*, [46, 90, 107]). However, in these models, time is treated as discrete index and not modeled as a random variable. In the diffusion network discovery problem, time is treated explicitly as a continuous variable, and one is interested in capturing how the occurrence of event at one node affects the time for its occurrence at other nodes. This problem recently has been explored by a number of studies in the literature. Specifically, Meyers and Leskovec inferred the diffusion network by learning the infection probability between two nodes using a convex programming, called CONNIE [120]. Gomez-Rodriguez et al. inferred the network connectivity using a submodular optimization, called NETINF [58]. However, both CONNIE and NETINF assume that the transmission model for each pair of nodes is fixed with predefined transmission rate. Somanchi et al. [151] propose a greedy algorithm to uncover the network structure by computing the highest-scoring connected subgraph. However, it cannot learn the temporal dynamics associated with each edge. Recently, Gomez-Rodriguez et al. proposed an elegant method, called NETRATE [56], using continuous temporal dynamics model to allow variable diffusion rates across network edges. NETRATE makes fewer number of assumptions and achieves better performance in various aspects than the previous two approaches. However, the limitation of NETRATE is that it requires the influence model on each




(a) Pair 1 (b) Pair 2 (c) Pair 3

Figure 3.1: The histograms of the interval between the time when a post appeared in one site and the time when a new post in another site links to it. Dotted and dash lines are density fitted by NetRate. The solid lines are given by KernelCascade.

edge to have a fixed parametric form, such as exponential, power-law, or Rayleigh distribution, although the model parameters learned from cascades could be different.

In practice, the patterns of information diffusion (or a spreading disease) among entities can be quite complicated and different from each other, going far beyond what a single family of parametric models can capture. One example is from the information diffusion in a blog-sphere: the hyperlinks between posts can be viewed as some kind of information flow from one media site to another, and the time difference between two linked posts reveal the pattern of diffusion. In Figure 3.1, we examined three pairs of media sites from the MemeTracker dataset [56, 99], and plotted the histograms of the intervals between the the moment when a post first appeared in one site and the moment when it was linked by a new post in another site. We can observe that information can have very different transmission patterns for these pairs. Parametric models fitted by NetRate may capture the simple pattern in Figure 3.1(a), but they might miss the multimodal patterns in Figure 3.1(b) and Figure 3.1(c). In contrast, our method, called KernelCascade, is able to fit both data accurately and thus can handle the heterogeneity.

The remainder of this chapter is organized as follows: in Section 3.2, we present our multivariate terminating point process for describing the continuous-time information diffusion process. In Section 3.3, we describe efficient learning algorithms with

Table 3.1: Table of symbols

| Symbol | Description |
|---|---|
| $\mathcal{G}$ | Directed network |
| $\mathcal{V}$ | Set of all the nodes in graph $\mathcal{G}$ |
| $\mathcal{E}$ | Set of all the edges in graph $\mathcal{G}$ |
| $f_{ji}(t)$ | Density of the diffusion time from node $j$ to $i$ |
| $\boldsymbol{t}^c$ | $N$-dimensional vector recording the infection time of each node in a cascade |
| $t_i^c$ | Infection time of node $i$ in the cascade $c$ |
| $\mathcal{C}$ | Collection of cascades |
| $C$ | Total number of cascades $C = \lvert\mathcal{C}\rvert$ |
| $\lambda_i^*(t)$ | Conditional intensity function of node $i$ |
| $\lambda_i^0$ | Base intensity function of node $i$ |
| $\phi_{ji}(t)$ | Infection risk function from node $j$ to $i$ |
| $\boldsymbol{m}_j^{t_j}$ | Topic vector of the meme published by node $j$ at the time $t_j$ |
| $N$ | Total number of nodes |
| $T_c$ | Observation window in the cascade $c$ |
| $\boldsymbol{\theta}_j$ | Parameters for the Dirichlet distribution for node $j$ |
| $\lambda$ | Regularization coefficient for the group-lasso |

elegant convex formulations for estimating the model parameters. In Section 3.4, we conduct experimental evaluations on both synthetic and real-world datasets. Finally, we summarize our contributions in Section 3.5. To facilitate the subsequent presentation, table 3.1 provides a list of symbols and their definitions used throughout this chapter.

## 3.2 Information Diffusion Model

Given a *directed* network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, our basic assumptions are that information propagates independently over the edges, and it takes certain amount of time for the information to spread from one node to another following the direction of the respective edge. The process begins with a set of infected source nodes, $\mathcal{A}$, initially adopting certain *contagion* (idea, meme or product) at time zero. The contagion is transmitted from the sources along their out-going edges to their direct neighbors.

Each transmission through an edge entails a *random* spreading time, $\tau$, drawn from a density over time, $f_{ji}(\tau)$. We assume transmission times are independent and possibly distributed differently across edges. Then, the infected neighbors transmit the contagion to their respective neighbors, and the process continues. We assume that an infected node remains infected for the entire diffusion process. Thus, if a node $i$ is infected by multiple neighbors, only the neighbor that first infects node $i$ will be the *true parent*. As a result, although the contact network can be an arbitrary directed network, each cascade (a vector of event timing information from the spread of a contagion) induces a Directed Acyclic Graph (DAG). This model is referred to as the continuous-time independent cascade model first studied in [56] by extending the discretized counterpart originally proposed in [86].

### 3.2.1 Model Formulation

Our input thus comprise only a collection $\mathcal{C}$, of cascades $\left\{\boldsymbol{t}^1, \ldots, \boldsymbol{t}^{|\mathcal{C}|}\right\}$, where each cascade is an $N$-dimensional vector $\boldsymbol{t}^c := (t_1^c, \ldots, t_N^c)^\top$ with $i$-th dimension recording the time stamp when event $c$ occurs to node $i$. Furthermore, $t_i^c \in [0, T^c] \cup \{\infty\}$, and the symbol $\infty$ labels nodes that have not been influenced during observation window $[0, T^c]$. The 'clock' is set to 0 at the start of each cascade. Based on the input cascade data $\mathcal{C}$, our output is the recovered hidden network structure with pairwise risk functions $\phi_{ji}$, which quantifies how fast the information can flow from node $j$ to node $i$.

The key idea to model the information diffusion process is to treat each user as a single dimension. Given $n$ users in total, we thus have a $n$-dimensional multivariate point process. Since we assume that each user keeps the state of infection after adopting a virus (or idea, product), in the current information diffusion setting, at most one *terminating* event (or infection) can happen to each dimension, which corresponds to the infection moment of the respective user. To accurately describe the

diffusion process, we propose the Multivariate Terminating Process, which is formally defined as:

**Definition 3.1** *The Multivariate Terminating Point Process is an n-dimensional temporal point process with the conditional intensity function of each dimension d is given by* $\lambda_d^*(t) = \mathbb{I}\{N_d(t) \leq 1\} \cdot g(t)$ *where* $N_d(t)$ *is the number of events on the dimension d,* $g(t)$ *is a non-negative function, and* $\mathbb{I}\{\cdot\}$ *is the indicator function.*

We first model the conditional intensity (infection risk) of each node as an additive function of the influence from other nodes by

$$\underbrace{\lambda_i^*(t)}_{\text{total infection risk}} = \sum_{j \neq i} \underbrace{\phi_{ji}(t - t_j)\mathbb{I}\{t > t_j\}}_{\substack{\text{infection risk} \\ \text{from others}}}. \tag{3.1}$$

Then, we can derive the joint log-likelihood of observing the collection of cascades $\mathcal{C}$ by using the decomposition property of multivariate point process as

$$\ell\left(\{t_i^c\}^{i=1,\dots,N}\right) = \sum_{i=1}^{N}\left(\log \lambda_i^*(t_i^c) - \int_0^{T_c} \lambda_i^*(t)dt\right), \tag{3.2}$$

where we simply sum up all individual log-likelihood of observing each single cascade $\boldsymbol{t}^c$. Finally, we can learn the pairwise infection risk function $\{\phi_{ji}(t)\}$ by solving the following optimization problem with properly chosen regularization.

$$\max_{\{\phi_{ji}\}} \frac{1}{C}\sum_{c=1}^{C}\sum_{i=1}^{N}\left(\log \lambda_i^*(t_i^c) - \int_0^{T_c} \lambda_i^*(t)dt\right) - \lambda \sum_i \sum_{j \neq i} \|\phi_{ji}\|, \tag{3.3}$$

where the overall recovered network structure is reflected by the nonzero patterns of the pairwise infection risk functions $\phi_{ji}(t)$.

One thing worth to note is that we can extend the intensity function of 3.1 by adding a baseline intensity $\lambda_i^0$ to be

$$\underbrace{\lambda_i^*(t)}_{\text{total infection risk}} = \lambda_i^0 + \sum_{j \neq i} \underbrace{\phi_{ji}(t)}_{\substack{\text{infection risk} \\ \text{from others}}}. \tag{3.4}$$

The baseline intensity $\lambda_i^0 \geq 0$ for each node captures the external influence from the outside world. For instance, people can get a piece of news from the mass media in addition to seeing it from the status updates of his (her) friends. From this perspective, when $\lambda_i^0 = 0$, the multivariate terminating point process incorporates the model formulation [56, 44] from the theory of survival analysis as one special case, which only focus on modeling the internal influence from the diffusion process captured by the pairwise infection risk function $\{\phi_{ji}(t)\}$.

### 3.2.2 Model Parametrization

Depending on the problem of interests, we have several ways to formulate the pairwise infection risk function $\phi_{ji}(t)$.

**Parametric formulation (TopicCascade).** In the simplest case, $\phi_{ji}(t) = \alpha_{ji}$ is a constant, which means the transmission time over the edge from node $j$ to $i$ has an exponential distribution. If $\phi_{ji}(t) = \alpha_{ji} \cdot t$ is a linear function of time $t$, then the respective transmission time has a Rayleigh distribution, which often better captures the phenomenon that there is a typical response time for posts of a particular topic, and it is less likely the response time is very different from the mode than the exponential distribution.

Alternatively, we can also make $\phi_{ji}(t) = \alpha_{ji} \cdot t$ depend on the contextual features of the transmitted information, such as the topic of the content. The motivation is that ideas, tweets, styles, and online advertisements of different topics might have different transmission speed even between the same pair of nodes. As a result, for Ad-providers, they would like to know how to make their advertisements reach a large number of target consumers in a very short term. To achieve this goal, they have to figure out quantitatively the spreading speed of different advertisements among different communities. Outdoor equipment promotions spread faster within a community of mountaineering club than within a political group. In contrast, messages

from political campaigns may instead diffuse faster in the latter group.

As a consequence, we can represent each piece of information (or meme) as a topic vector in the canonical $K$-dimensional simplex, in which each component is the weight of a topic. Suppose that node $j$ just published a meme $\mathbf{m}_j^{t_j}$ at time $t_j$. We can represent it as $\mathbf{m}_j^{t_j} := (m_1, \ldots, m_K)^\top$ and $\sum_i m_i = 1$ and $m_i \in [0, 1]$. Such a representation can be readily obtained from the text of a meme using standard topic model tools such as Latent Dirichlet Allocation [20]. Then, to incorporate the topic information, $\mathbf{m}_j^{t_j}$, we assume that $\alpha_{ji}$ is a nonnegative combination of the entries in $\mathbf{m}_j^{t_j}$, that is,

$$\alpha_{ji} = \sum_{l=1}^{K} \alpha_{ji}^l \cdot m_l, \tag{3.5}$$

where $\alpha_{ji}^l \geqslant 0$ ensures that the risk function $\phi_{ji}(t) = \sum_{l=1}^{K} \alpha_{ji}^l m_l \cdot t$ is nonnegative. We call this type of formulation as the topic-modulated Rayleigh distribution, and refer to the respective diffusion model as the TopicCascade.

**Non-Parametric formulation (KernelCascade).** In real world applications, since the true distribution for the pairwise transmission time is never known, it is also better to formulate $\phi_{ji}(t)$ non-parametrically to capture the heterogeneous transmission patterns over the edges. Therefore, we can kernelize $\phi_{ji}(t)$, by assuming that it is a linear combination of $m$ kernel functions, *i.e.*,

$$\phi_{ji}(t) = \sum_{l=1}^{m} \alpha_{ji}^l \cdot k(\tau_l, t), \tag{3.6}$$

where we fix one argument of each kernel function, $k(\tau_l, \cdot)$, to a point $\tau_l$ in a uniform grid of $m$ locations in the range of $(0, \max_c T^c]$. To achieve fully nonparametric modeling of the hazard function, we can let $m$ grow as we see more cascades. In the formulation, we need to perform integration over the kernel function in order to

evaluate the integral of the intensity function $\int_0^t \lambda^*(\tau)d\tau$ in the log-likelihood 3.2 by

$$\int_0^t \lambda_i^*(\tau)d\tau = \sum_{j\neq i}\sum_{l=1}^{m} \alpha_{ji}^l \int_0^t k(\tau_l, \tau)d\tau. \tag{3.7}$$

Fortunately, this can be done efficiently for many kernels, such as the Gaussian RBF kernel, the Laplacian kernel, the Quartic kernel, and the Triweight kernel. In later experiments, we mainly focus on the Gaussian RBF kernel, $k(\tau_l, \tau_s) = \exp(-\|\tau_l - \tau_s\|^2/(2\sigma^2))$, and derive a closed form solution as

$$\int_0^t k(\tau_l, \tau)\ d\tau = \frac{\sqrt{2\pi}\sigma}{2}\left(\operatorname{erfc}\left(\frac{\tau_l - t}{\sqrt{2}\sigma}\right) - \operatorname{erfc}\left(\frac{\tau_l}{\sqrt{2}\sigma}\right)\right), \tag{3.8}$$

where $\operatorname{erfc}(t) := \frac{2}{\sqrt{\pi}}\int_t^\infty e^{-x^2}dx$ is the error function. Yet, our method is not limited to the particular RBF kernel. If there is no closed form solution for the one-dimensional integration for this purpose [129]. We note that given a dataset, both the kernel functions $\{k(\tau_l, \tau)\}_{l=1}^m$ and the respective integral $\left\{\int_0^t k(\tau_l, \tau)d\tau\right\}_{l=1}^m$ need to be computed only once as a preprocessing, and then can be reused in the learning algorithm later.

Besides, we can also place a non-linear basis function on each time point in the observed cascades. For efficiency consideration, we will use a fixed uniform grid in our later experiments. A nice property of the formulation 3.3 is that it is a convex function with respect to the parameters $\{\alpha_{ji}\}$ if we parametrize the risk functions with 3.5 and 3.6. As a consequence, it allows us to bring various efficient convex optimization techniques later to achieve the global solution efficiently. We refer to this type of non-parametric formulation as the KernelCascade.

## 3.3 Efficient Learning Algorithm

For uncovering the latent interdependency structure, we have the following convex formulation

$$\max_{\{\phi_{ji}\}} \frac{1}{C}\sum_{c=1}^{C}\sum_{i=1}^{N}\left(\log \lambda_i^*(t_i^c) - \int_0^{T_c} \lambda_i^*(t)dt\right) - \lambda \sum_i \sum_{j\neq i} \|\phi_{ji}\|,$$

where $\phi_{ji}$ is parametrized by a *vector of parameters* $\boldsymbol{\alpha}_{ji}$. Basically, if the coefficients $\boldsymbol{\alpha}_{ji} = \mathbf{0}$, then there is no edge (or direct influence) from node $j$ to $i$. For this purpose, we will impose grouped lasso type of regularization on the coefficients $\boldsymbol{\alpha}_{ji}$, *i.e.*, $(\sum_j \|\boldsymbol{\alpha}_{ji}\|_2)$ [133, 169]. Grouped lasso type of regularization has the tendency to select a small number of groups of non-zero coefficients but push other groups of coefficients to be zero. Therefore, we derive the following optimization problem that trades off between data likelihood term and group sparsity of the coefficients.

$$\min_{\{\boldsymbol{\alpha}_{ji}\}} -\left(\frac{1}{C}\sum_{c=1}^{C}\sum_{i=1}^{N}\left(\log \lambda_i^*(t_i^c) - \int_0^{T_c} \lambda_i^*(t)dt\right) - \lambda \sum_i \sum_{j\neq i} \|\boldsymbol{\alpha}_{ji}\|_2\right),$$
$$\text{subject to } \boldsymbol{\alpha}_{ji} \geq 0, \quad \forall j \in \{1, \ldots, N\}, \tag{3.9}$$

where we minimize the negative log-likelihood subject to the non-negative constraint. First note that the optimization 3.9 is separable with respect to each individual node $i$, so we can optimize with respect to each whole column $\boldsymbol{\alpha}_{:,i}$ which centers around the node $i$ in parallel. We can then join all the partial structures together and obtain the overall diffusion network. Second, the optimization problem in (3.9) is a non-smooth but separable minimization problem, which can be difficult to optimize using standard methods. We will use the following block coordinate descent framework [158] to efficiently find a solution. Let the log-likelihood terms associated with node $i$ be defined as

$$\mathcal{L}_i\left(\{\boldsymbol{\alpha}_{ji}\}_{j=1}^N\right) := -\frac{1}{C}\sum_{c=1}^{C}\left(\log \lambda_i^*(t_i^c) - \int_0^{T_c} \lambda_i^*(t)dt\right). \tag{3.10}$$

At each iteration, we solve the problem (3.9) with respect to one $\boldsymbol{\alpha}_{ji}$ variable while fixing other variables, *i.e.*,

$$\begin{cases} \text{For } k = 1, \ldots, k_{\max} \\ \quad \boldsymbol{\alpha}_{1i}^{k+1} = \underset{\boldsymbol{v}_1 \geq \mathbf{0}}{\operatorname{argmin}} \ \ell_{1i}(\boldsymbol{v}_1) + \lambda \left\| \boldsymbol{v}_1 \right\|_2 \\ \quad \boldsymbol{\alpha}_{2i}^{k+1} = \underset{\boldsymbol{v}_2 \geq \mathbf{0}}{\operatorname{argmin}} \ \ell_{2i}(\boldsymbol{v}_2) + \lambda \left\| \boldsymbol{v}_2 \right\|_2 \\ \quad \cdots \\ \quad \boldsymbol{\alpha}_{Ni}^{k+1} = \underset{\boldsymbol{v}_N \geq \mathbf{0}}{\operatorname{argmin}} \ \ell_{Ni}(\boldsymbol{v}_N) + \lambda \left\| \boldsymbol{v}_N \right\|_2 \end{cases} \tag{3.11}$$

where the function $\ell_{ji}(\boldsymbol{v}_j)$ is defined by fixing all other coordinates to their current values in the iteration, *i.e.*,

$$\ell_{ji}(\boldsymbol{v}_j) := \mathcal{L}_i \left( \boldsymbol{\alpha}_{1i}^{k+1}, ..., \boldsymbol{\alpha}_{(j-1)i}^{k+1}, \boldsymbol{v}_j, \boldsymbol{\alpha}_{(j+1)i}^{k}, ..., \boldsymbol{\alpha}_{Ni}^{k} \right).$$

Essentially, the optimization is carried out by solving a sequence of subproblems each involving only one $\boldsymbol{\alpha}_{ji}$. Since $\mathcal{L}_i \left( \{\boldsymbol{\alpha}_{ji}\}_{j=1}^N \right)$ is sufficiently smooth convex fitting terms and $\sum_j \left\| \boldsymbol{\alpha}_{ji} \right\|_2$ is non-differentiable but separable regularization terms, the global convergence of the above algorithm is guaranteed by results from [158].

Unfortunately, it can still be difficult to find a solution for each subproblem directly due to the non-smooth regularizer. Thus we will use a proximal gradient method for these subproblems (*e.g.*, [17]). We apply an iterative procedure to solve each subproblem

$$\boldsymbol{\alpha}_{ji}^{k+1} = \underset{\boldsymbol{v}_j \geq \mathbf{0}}{\operatorname{argmin}} \ \mathcal{L}_{ji}(\boldsymbol{v}_j) + \lambda \left\| \boldsymbol{v}_j \right\|_2,$$

and more specifically, we will minimize a sequence of quadratic approximation to $\mathcal{L}_{ji}(\boldsymbol{v}_j)$ by

$$\mathcal{L}_{ji}^s(\boldsymbol{v}_j, \boldsymbol{v}_j') = \mathcal{L}_{ji}(\boldsymbol{v}_j') + \left\langle \nabla \mathcal{L}_{ji}(\boldsymbol{v}_j'), \boldsymbol{v}_j - \boldsymbol{v}_j' \right\rangle + \frac{1}{2} \left\langle \boldsymbol{v}_j - \boldsymbol{v}_j', \boldsymbol{D}_{\boldsymbol{v}_j'}(\boldsymbol{v}_j - \boldsymbol{v}_j') \right\rangle,$$

where $\boldsymbol{D}_{\boldsymbol{v}_j'}$ is a positive definite matrix that dominates the Hessian $\frac{\partial^2 \mathcal{L}_{ji}(\boldsymbol{v}_j')}{\partial (\boldsymbol{v}_j')^2}$. This surrogate function has the following useful properties: $\mathcal{L}_{ji}^s(\boldsymbol{v}_j, \boldsymbol{v}_j) = \mathcal{L}_{ji}(\boldsymbol{v}_j)$ and

$\mathcal{L}^s_{ji}(\boldsymbol{v}_j, \boldsymbol{v}'_j) \geqslant \mathcal{L}_{ji}(\boldsymbol{v}_j)$, $\forall \boldsymbol{v}, \boldsymbol{v}'_j \geq \mathbf{0}$. As a consequence, we find $\boldsymbol{\alpha}^{k+1}_{ji}$ by the following iterative procedure

$$\begin{cases} \boldsymbol{v}'_j \leftarrow \boldsymbol{\alpha}^k_{ji} \\ \text{For } l = 1, \ldots, l_{\max} \\ \quad \boldsymbol{v}'_j \leftarrow \underset{\boldsymbol{v}_j \geq 0}{\operatorname{argmin}} \ \mathcal{L}^s_{ji}(\boldsymbol{v}_j, \boldsymbol{v}'_j) + \lambda \|\boldsymbol{v}_j\|_2 \\ \boldsymbol{\alpha}^{k+1}_{ji} \leftarrow \boldsymbol{v}'_j \end{cases} \tag{3.12}$$

If $\boldsymbol{D}_{\boldsymbol{\alpha}^k_{ji}} = L \cdot \boldsymbol{I}$, then we only need to solve the following proximal mapping

$$\boldsymbol{v}'_j \leftarrow \underset{\boldsymbol{v}_j \geq 0}{\operatorname{argmin}} \frac{L}{2} \left\| \boldsymbol{v}_j - \left( \boldsymbol{v}'_j - \frac{\nabla \mathcal{L}_{ji}(\boldsymbol{v}'_j)}{L} \right) \right\|^2_2 + \lambda \|\boldsymbol{v}_j\|_2,$$

which has a closed form solution

$$\boldsymbol{v}'_j \leftarrow \left( \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2} \left( \|\boldsymbol{v}\|_2 - \frac{\lambda}{L} \right)_+ \right)_+, \tag{3.13}$$

where $\boldsymbol{v} = \boldsymbol{v}'_j - \frac{1}{L} \nabla \mathcal{L}_{ji}(\boldsymbol{v}'_j)$ and $(\cdot)_+$ simply sets the negative coordinates of its argument to 'zero'. If $\boldsymbol{v}'_j = \mathbf{0}$ at some point in the iteration, we just stop updating it and directly assign zeros to $\boldsymbol{\alpha}^{k+1}_{ji}$. The learning rate is often set to $1/L$ accordingly.

The overall pseudo codes are given in Algorithm 3.1. In addition, because the optimization is independent for each node $i$, the outer loop process can be easily parallelized into $N$ separate sub-problems. Meanwhile, we can prune the possible nodes that never appeared before node $i$ in any cascade indicating that $j$ is not a possible parent of $i$ that is at least shown from the data. In the end, if we further assume that all the edges from the same node have similar temporal dynamics (or similar topics), especially when the sample size is small, the $N$ edges could share a common set of $K$ parameters, and we can only estimate $N \times K$ parameters in total. By exploiting $P$ cores, the computational complexity is $O(CN^2K/P)$.

## 3.4 Experiments

We evaluate TopicCascadeand KernelCascade on both realistic synthetic networks and real-world networks. We compare them to the state-of-the-art network

**Algorithm 3.1:** Structure Learning

**Input**: cascades $\left\{\boldsymbol{t}^1, ..., \boldsymbol{t}^{|\mathcal{C}|}\right\}$
**Output**: $\{\boldsymbol{\alpha}_{ji}\}_{j,i\in[1,N]}$

```
for i = 1 to N do
    Initialize {α_ji}_{j=1}^N as uniform vector;
    repeat
        Update {α_ji}_{j=1}^N using the formula (3.11) and (3.12) from last values;
    until convergence;
    Extract the sparse neighborhood 𝒩(i) of node i from nonzero α_ji;
end
```

structure learning methods NetRate [56], NetInf [58], and MoNET [164] showing that TopicCascade and KernelCascade can perform significantly better in terms of both recovering the network structures and predicting the transmission time.

### 3.4.1 TopicCascade on Synthetic Data

**Network Generation.** We generate synthetic networks that mimic the structural properties of real networks. These synthetic networks can then be used for simulation of information diffusion. Since the latent networks for generating cascades are known in advance, we can perform detailed comparisons between various methods. We use Kronecker generator [102] to examine three types of networks with directed edges: (i) the core-periphery structure [101] with parameters [0.9 0.5; 0.5 0.3], which mimics the information diffusion traces in real world networks, (ii) the Erdős-Rényi random networks with parameters [0.5 0.5; 0.5 0.5], being typical in physics and graph theory, and (iii) the hierarchy networks with parameters [0.9 0.1; 0.1 0.9].

**Topic Generation.** Each node $j$ is assigned a uniformly distributed random variable $\boldsymbol{\theta}_j \in (0, 1]^K$ which is the parameter for a $K$-dimensional symmetric Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\theta}_j)$. Then we can sample a $K$-dimensional topic distribution $\boldsymbol{m}_j$ from $\mathrm{Dir}(\boldsymbol{\theta}_j)$ for each node $j$. Since the entries of $\boldsymbol{\theta}$ is less than one, the generated memes are more focused on a small subsets of topics.

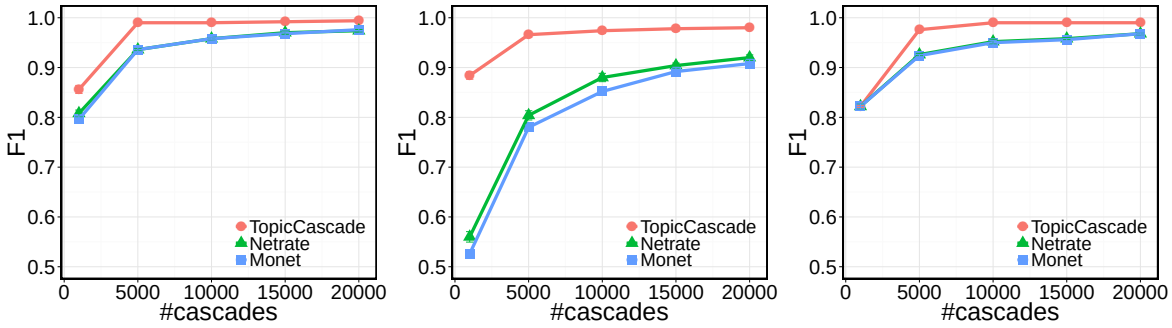
**Cascade Generation.** For each pair of nodes $j$ and $i$, the edge $j \to i$ is randomly assigned a $K$-dimensional topic preference vector $\boldsymbol{\alpha}_{ji}$ where each component $\alpha_{ji}^l \in [0, 1]$. Given a network $\mathcal{G}$, we generate a cascade from $\mathcal{G}$ by randomly choosing a node $j$ as the root of the cascade. The root node $j$ is then assigned to time stamp $t_j = 0$. We sample a meme $\boldsymbol{m}_j^{t_j}$ from $\mathrm{Dir}(\theta_j)$. Then, for each neighbor node $i$ pointed by $j$, its event time $t_i$ given $t_j$ is sampled from the topic-modulated Rayleigh distribution

$$f_{ji}^m(\Delta_{ji}) = \boldsymbol{\alpha}_{ji}^\top \mathbf{m}_j^{t_j} \Delta_{ji} \exp\left(-\frac{1}{2}\boldsymbol{\alpha}_{ji}^\top \mathbf{m}_j^{t_j} \Delta_{ji}^2\right), \tag{3.14}$$

where $\Delta_{ji} = t_i - t_j$. The child node $i$ will copy the received meme $\boldsymbol{m}_j^{t_j}$ and forward it to its own children. In general, node $i$ can add a small disturbance to the vector $\boldsymbol{m}_j^{t_j}$ to generate a slightly different meme. However, for simplicity, in our later experiments, we assume that node $i$ directly copies $\boldsymbol{m}_j^{t_j}$ as its own meme. The diffusion process will continue by further infecting the neighbors pointed by node $i$ in a breadth-first fashion until either the overall time exceed the predefined observation time window $T^c$, or there is no new node being infected. If a node is infected more than once by multiple parents, only the first infection time stamp and the meme will be recorded.

**Experimental Setting.** For each type of synthetic networks, we randomly instantiate the network topologies and all the required parameters ( $\boldsymbol{\alpha}_{ji}$ for each edge and the set of memes $\boldsymbol{m}_j^{t_j}$) for five times. The number of cascades varies from 1000, 5000, 10000, 15000 to 20000. For each experiment setting, the regularization parameter is chosen based on two-fold cross validation, and the experimental results are reported on a separate hold-out test set consisting of the same number of cascades. For NetRate and MoNET, we fit them with a Rayleigh transmission model.

**Evaluation Metrics.** We have considered three different metrics: (1) we first compare the $F1$ score for the network recovery. $F1 := \frac{2 \cdot precision \cdot recall}{precision + recall}$, where *precision* is the fraction of edges in the inferred network that also present in the true network



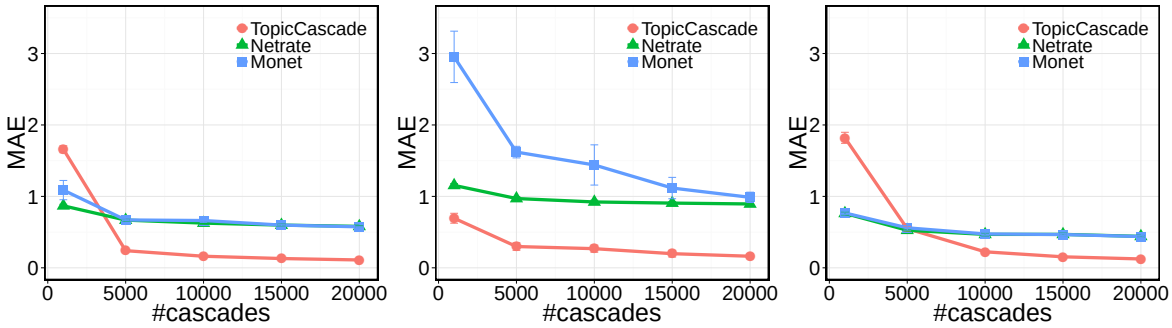

(a) Random (b) Core-Periphery (c) Hierarchy

Figure 3.2: F1 scores for network recovery. Each network has 512 nodes and 1024 edges.

and *recall* is the fraction of edges in the true network that also present in the inferred network; (2) we compare the modes of the fitted Rayleigh distributions given by TopicCascade, NetRate, and MoNET with the true mode for each meme $\boldsymbol{m}_j$ from $j \to i$, and report the Mean Absolute Error (MAE). (3) and finally, we report the distance $\|\hat{\boldsymbol{\alpha}}_{ji} - \boldsymbol{\alpha}_{ji}\|_2$ between the estimated parameters and the true ones as number of cascades varies.

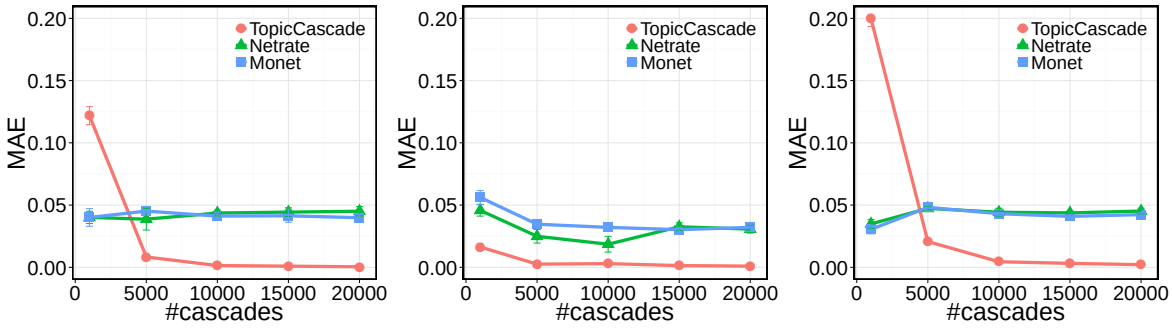
**F1 score for network recovery.** From Figure 3.2, we observe in all cases, TopicCascade performs consistently and significantly better than NetRate and MoNET. Furthermore, its performance also steadily increases as we increase the number of cascades, and finally TopicCascade almost recovers the entire network with around 10000 cascades. In contrast, the competitor method seldom fully recovers the entire network given the same number of cascades, although it has less parameters to fit.

**MAE for mode prediction.** Figure 3.3 presents the MAE between the estimated mode and the true mode. When there are only 1,000 cascades, NetRate and MoNET can perform better than TopicCascade, since TopicCascade has more parameters to estimate. However, as the number of cascades grows, the MAE for TopicCascade quickly decreases. In contrast, NetRate and MoNET keep a flat error rate since they do not take the topics into account.




(a) Random (b) Core-Periphery (c) Hierarchy

Figure 3.3: Absolute mean errors between the estimated modes and the true modes.




(a) Random (b) Core-Periphery (c) Hierarchy

Figure 3.4: Relative errors between the estimated and the true median mode.

To further illustrate the extent to which the estimated mode is close to the true value, we plot the relative error of the estimated median mode with respect to the true value in Figure 3.4. In all cases, the estimation given by TopicCascade goes to the true mode quickly as the number of cascades increases, while NetRate and MoNET keep an almost steady error rate.

**Distance between the estimation and the true value.** Because in our simulations we have known in advance the true topic preference parameter on each edge, we would like to check how close the estimated $\hat{\boldsymbol{\alpha}}_{ji}$ is close to the true $\boldsymbol{\alpha}_{ji}$. In Figure 3.5, we plot the average distance between $\hat{\boldsymbol{\alpha}}_{ji}$ and $\boldsymbol{\alpha}_{ji}$. Again, it shows that $\hat{\boldsymbol{\alpha}}_{ji}$ approaches to $\boldsymbol{\alpha}_{ji}$ quickly as the number of cascades increases.

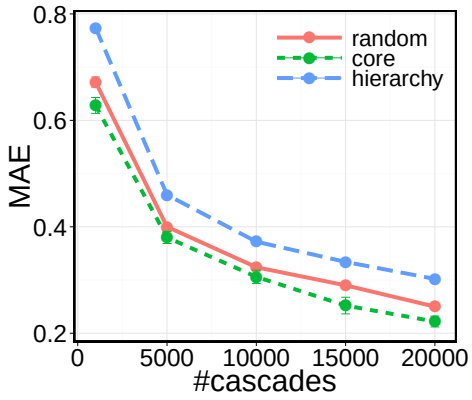



Figure 3.5: Mean absolute error decreases as we use more training cascades.

### 3.4.2 TopicCascade on Real Data

We use the MemeTracker dataset [99] to evaluate our model. In this dataset, the hyperlinks between articles and posts can be used to represent the flow of information from one site to another site. When a site publishes a new post, it will put hyperlinks to related posts in some other sites published earlier as its sources. Later as it also becomes "older", it will be cited by other newer posts as well. As a consequence, all the time-stamped hyperlinks form a cascade for particular piece of information (or event) flowing among different sites. The networks formed by these hyperlinks are used to be the ground truth. In addition, each post also has some texts to describe its content. There are totally 3,771 posts with 85,671 words. We have extracted a network consisting of top 500 sites with about 1,100 edges and 2,000 cascades. The maximum time is 614 hours. We first fit an LDA [20] model to the corpus of posts and extract ten typical topics shown in Table 3.2 where each topic includes the top four descriptive words.

For each pair of nodes $i$ and $j$, given a post topic $\boldsymbol{m}_j$, we use the modes of the fitted Rayleigh distributions as the estimated transmission time from $j$ to $i$ for the post $\boldsymbol{m}_j$. We then report the MAE and the median error between the estimated time and the real time on the correctly estimated edges in Table 3.3. We start to compare different models first by using the whole dataset to train and report the different

Table 3.2: Ten topics learned from the posts.

| | | | | |
|---|---|---|---|---|
| Topic 0: | information | social | computer | google |
| Topic 1: | people | women | life | god |
| Topic 2: | time | love | people | life |
| Topic 3: | market | year | money | economy |
| Topic 4: | war | georgia | military | country |
| Topic 5: | obama | mccain | president | campaign |
| Topic 6: | energy | oil | gas | plan |
| Topic 7: | people | laws | issues | free |
| Topic 8: | loss | men | till | las |
| Topic 9: | windows | web | technology | games |

Table 3.3: Estimations in the MemeTracker dataset with Rayleigh transmission functions on the correctly predicted edges.

| Models | Train | | | Test | |
|---|---|---|---|---|---|
| | F1 | MAE | Median | MAE | Median |
| TopicCascade | 0.60 | 15.8 | 7.4 | 28.0 | 2.4 |
| NetRate | 0.23 | 31.8 | 21.2 | 33.7 | 24.9 |
| MoNET | 0.14 | 32.3 | 20.8 | 35.2 | 25.8 |

Table 3.4: Estimations in the MemeTracker dataset with exponential transmission functions on the correctly predicted edges.

| Models | Train | | | Test | |
|---|---|---|---|---|---|
| | F1 | MAE | Median | MAE | Median |
| TopicCascade | 0.72 | 174 | 14.9 | 241 | 9.3 |
| NetRate | 0.81 | 1307 | 532.8 | 538.5 | 27.1 |
| MoNET | 0.72 | 1317 | 527 | 585 | 27 |

metrics in the *train* column of Table 3.3. Then, we uniformly select 10-percent of the edges as the test data and use the other 90-percent data to train. The experiments have been repeated for 10 times. We report the average value of all the metrics in the *test* column of Table 3.3. Moreover, we also repeat all the above experiments by fitting a topic-modulated exponential transmission function for each edge and use the expectation as the estimated time on each correctly predicted edge in Table 3.4. In all cases, TopicCascade have lower MAE, Median error as well as a relatively better F1 score. For both NetRate and MoNET, they are also fitted by the exponential

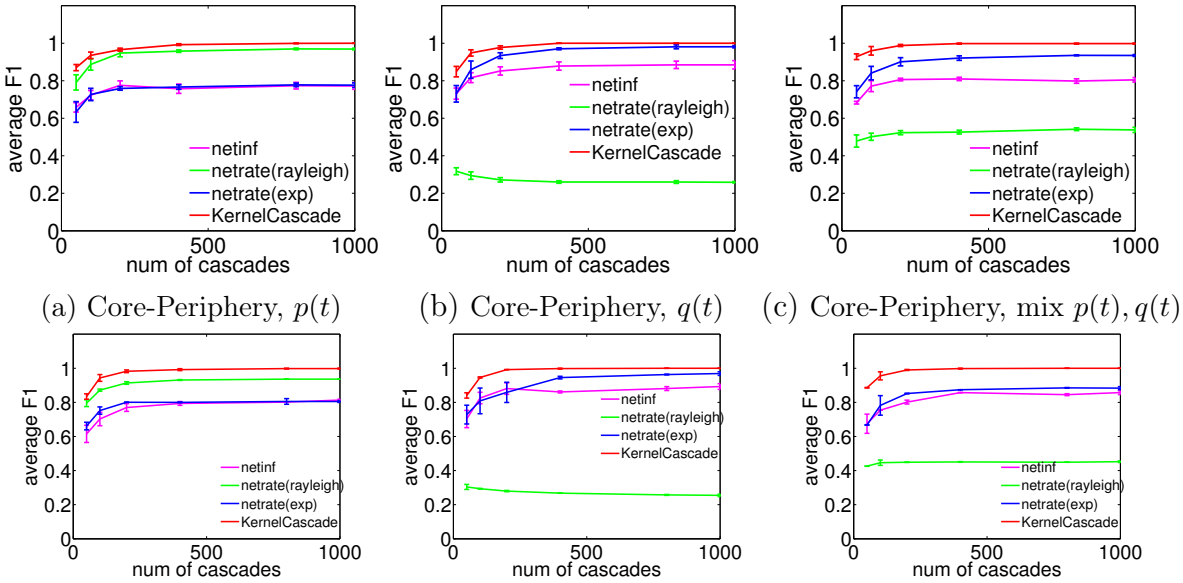
distribution to the MemeTracker dataset in order to recover the network structure. Because the exponential model is relatively easier to fit than the Rayleigh distribution, it gives a better F1 score. However, when we use the expectation of the exponential to predict the time, Table 3.4 shows that it has large MAE and Median error in terms of predicting the actual time of the events.

### 3.4.3 KernelCascade on Synthetic Data

**Influence function.** We first generate the synthetic networks with the Kronecker generator [102] as before. Then, for each edge $j \to i$ in a network $\mathcal{G}$, we will assign it a mixture of two Rayleigh distributions: $f_{ji}(t|\theta, a_1, b_1, a_2, b_2) = \theta R_1(t|a_1, b_1) + (1-\theta)R_2(t|a_2, b_2)$ where $R_i(t|a_i, b_i) = \frac{2}{t-a_i}\left(\frac{t-a_i}{b_i}\right)^2 \exp\left(-\left(\frac{t-a_i}{b_i}\right)^2\right)$, $t \geqslant a_i$, and $\theta \in (0, 1)$ is a mixing proportion. We examine three different parameter settings for the spreading time distribution (or transmission function): (1) all edges in network $\mathcal{G}$ have the same transmission function $p(t) = f(t|0.5, 10, 1, 20, 1)$; (2) all edges in network $\mathcal{G}$ have the same transmission function $q(t) = f(t|0.5, 0, 1, 20, 1)$; and (3) all edges in network $\mathcal{G}$ are uniformly randomly assigned to either $p(t)$ or $q(t)$.

**Cascade generation.** Given a network $\mathcal{G}$ and the collection of transmission functions $f_{ji}$ for each edge, we generate a cascade from $\mathcal{G}$ by randomly choosing a node of $\mathcal{G}$ as the root of the cascade. The root node $j$ is then assigned to time stamp $t_j = 0$. For each neighbor node $i$ pointed by $j$, its event time $t_i$ is sampled from $f_{ji}(t)$. The diffusion process will continue by further infecting the neighbors pointed by node $i$ in a breadth-first fashion until either the overall time exceed the predefined observation time window $T^c$ or there is no new node being infected. If a node is infected more than once by multiple parents, only the first infection time stamp will be recorded.

**Experiment setting and evaluation metric.** We consider a combination of two network topologies (i)-(ii) with three different transmission function settings (1)-(3),



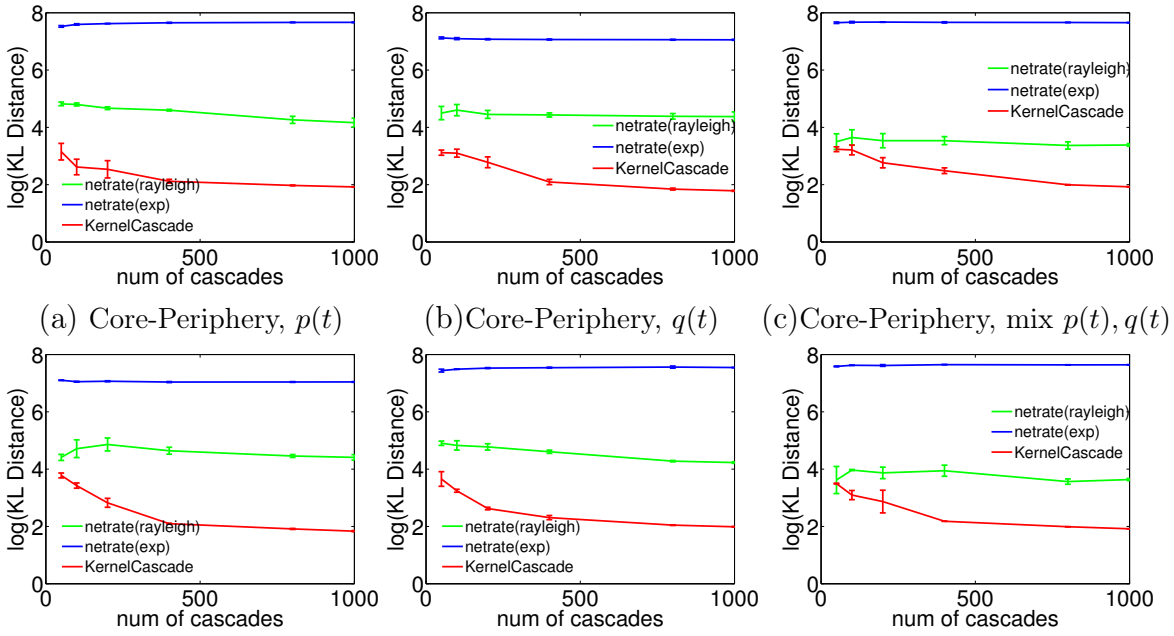

(a) Core-Periphery, $p(t)$ (b) Core-Periphery, $q(t)$ (c) Core-Periphery, mix $p(t), q(t)$

(d) Random, $p(t)$ (e) Random, $q(t)$ (f) Random, mix $p(t), q(t)$

Figure 3.6: F1 Scores for network recovery.

which results in six different experimental settings. For each setting, we randomly instantiate the network topologies and transmission functions for 10 times and then vary the number of cascades from $50, 100, 200, 400, 800$ to 1000. For KernelCascade, we use a Gaussian RBF kernel. The kernel bandwidth $\sigma$ is chosen using median pairwise distance between grid time points. The regularization parameter is chosen using two fold cross-validation. NetInf requires the desired number of edges as input, and we give it an advantage and supply the true number of edges to it. For NetRate, we experimented with both exponential and Rayleigh transmission function.

We compare different methods in terms of (1) $F1$ score for the network recovery. $F1 := \frac{2 \cdot precision \cdot recall}{precision + recall}$, where *precision* is the fraction of edges in the inferred network that also present in the true network and *recall* is the fraction of edges in the true network that also present in the inferred network; (2) $KL$ divergence between the estimated transmission function and the true transmission function, averaged over all edges in a network; (3) the shape of the fitted transmission function compared to the true transmission function.



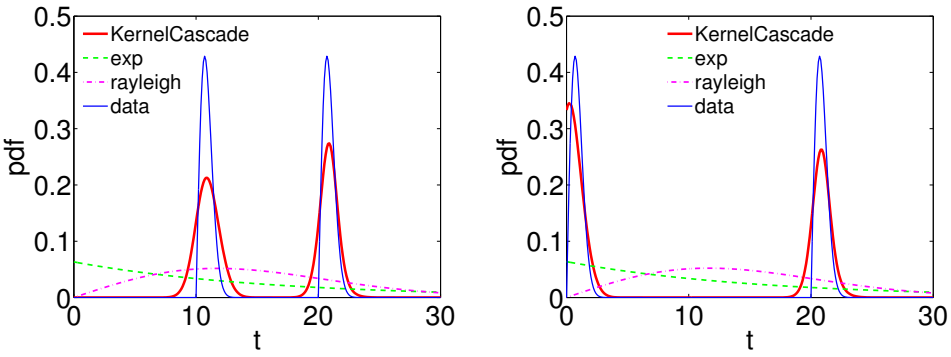

(a) Core-Periphery, $p(t)$ (b)Core-Periphery, $q(t)$ (c)Core-Periphery, mix $p(t), q(t)$

(d)Random, $p(t)$ (e)Random, $q(t)$ (f)Random, mix $p(t), q(t)$

Figure 3.7: KL Divergence between the estimated and the true transmission function.

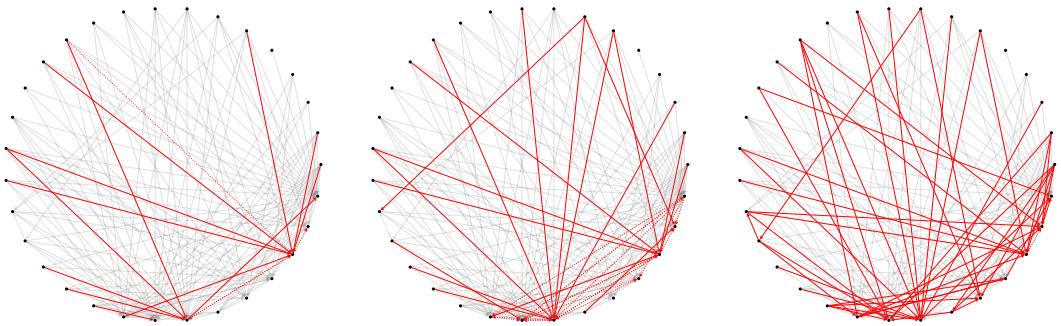
**F1 score for network recovery.** From Figure 3.6, we can see that in all cases, KernelCascade performs consistently and significantly better than NetInf and NetRate. Furthermore, its performance also steadily increases as we increase the number of cascades, and finally KernelCascade recovers the entire network with around $1,000$ cascades. In contrast, the competitor methods seldom fully recover the entire network given the same number of cascades. We also note that the performance of NetRate is very sensitive to the choice of the transmission function (exponential vs. Rayleigh). Depending on the specific data generating process, the performance of NetRate with Rayleigh model can vary from the second best to the worst.

**KL divergence of transmission function.** Besides better network recovery, KernelCascade also estimates the transmission function better. In all cases we experimented, KernelCascade leads to drastic improvement in recovering the transmission function (Figure 3.7). We also observe that as we increase the number of cascades, KernelCascade adapts better to the actual transmission function. In




(a) Transmission function $p(t)$ (b) Transmission function $q(t)$

Figure 3.8: Estimated transmission function of a single edge based on 1000 cascades against the true transmission function (blue curve).



(a) KernelCascade (b) NetInf (c) NetRate

Figure 3.9: Estimated network of top 32 sites. Edges in grey are correctly uncovered, while edges highlighted in red are either missed or estimated falsely.

contrast, the performance of NetRate with exponential model does not improve with increasing number of cascades, since the parametric model assumption is incorrect. We note that NetInf does not recover the transmission function, and hence there is no corresponding curve in the plot.

**Visualization of transmission function.** We also visualize the estimated transmission function for an edge from different methods in Figure 3.8. We can see that KernelCascade captures the essential features of the true transmission function, *i.e.*, bi-modal behavior, while the competitor methods miss out the important statistical feature completely.

Table 3.5: Network recovery results from MemeTracker dataset.

| Methods | Precision | Recall | F1 | #Predicted Edges |
|---|---|---|---|---|
| NetInf | 0.62 | 0.62 | 0.62 | 6466 |
| NetRate(exp) | 0.93 | 0.23 | 0.37 | 1600 |
| KernelCascade | 0.79 | 0.66 | **0.72** | 5368 |

### 3.4.4 KernelCascade on Real Data

We use the MemeTracker dataset [99] to compare NetInf, NetRate and KernelCascade. The networks formed by the hyperlinks are used to be the ground truth. We have extracted a network consisting of the top 500 sites with 6,466 edges and 11,530 cascades from 7,181,406 posts in a month. Table 3.5 shows that KernelCascade achieves a much better $F1$ score for network recovery compared to other methods. Finally, we visualize the estimated sub-network structure for the top 32 sites in Figure 3.9. By comparison, KernelCascade has a relatively better performance with fewer misses and false predictions.

## 3.5 Summary

Diffusions take place among networked entities due to complex collective interactions. Quite often, the underlying transmission networks are hidden, and we observe only the time stamps when sequences of events happen. Besides, the pairwise temporal diffusion dynamics are heterogeneous showing multi-modal characteristics and depending on many contextual factors, such as the topic of the diffusion. We have developed a multivariate terminating process to describe the diffusion processes over networks. The model only assumes infections propagate independently over the edges, and each node keeps the infection state thereafter (*e.g.* adopting an idea, forwarding a post, purchasing a product, *etc.*). In contrast to previous state-of-the-arts [58, 56, 164], our model does not have to make restricted parametric assumptions of the pairwise transmission functions. Instead, it can infer them adaptively from the data, which allows each pair of nodes to have a different type of transmission function and better

captures the heterogeneous influence among entities. Furthermore, it is also very easy to incorporate the possible effects of external influences into the model, which extends and includes the existing theories [58] based on the survival analysis as a special case.

We have proposed an efficient learning algorithm based on elegant convex formulations and a group-lasso type of regularization to obtain a sparse group-structure for the vectorized model parameters. By exploiting the specific structure of the optimization problem, our algorithm can be fully parallelized without any communication cost. Experimental results on both synthetic and real data demonstrate that our methods achieve significantly better performance compared to other state-of-the-arts.

Since that we have a better understanding of the latent diffusion process, can we make use of the extracted knowledge to optimize the diffusion process so as to generate social influence, improve user engagement and achieve desirable outcomes among individuals and organizations? In the next chapter, we move on to solve downstream inference tasks based on our learned models.

# CHAPTER IV

# SCALABLE INFLUENCE ESTIMATION IN CONTINUOUS-TIME DIFFUSION NETWORKS

Given that we have successfully learned the dependency structure and the associated temporal dynamics based on the proposed multivariate terminating process for the continuous-time information diffusion, our next goal is to make use of the extracted insights for making time-sensitive decisions. For instance, if a piece of information is released from a media site, can we predict whether it may spread to one million web pages, in a month? In addition, can we seed a few websites such that the spreading of the information can be maximized as far as possible within the given time window? These two problems are often referred to as the influence estimation and maximization problem respectively, which are very challenging since both the time-sensitive nature of the problems and the issue of scalability need to be addressed at the same time.

In this chapter, we illustrate the inference capability of our proposed framework by efficiently solving the influence estimation and maximization problems in continuous-time diffusion networks. More specifically, we propose a randomized algorithm for influence estimation in continuous-time diffusion networks. Our algorithm can estimate the influence of every node in a network with $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges to an accuracy of $\epsilon$ using $n = O(1/\epsilon^2)$ randomizations and up to logarithmic factors $O(n|\mathcal{E}| + n|\mathcal{V}|)$ computations. When used as a subroutine in a greedy influence maximization approach, our proposed algorithm is guaranteed to find a set of $C$ nodes with the influence of at least $(1 - 1/e)\,\mathrm{OPT} - 2C\epsilon$, where OPT is the optimal value. Experiments on both synthetic and real-world data show that the proposed algorithm can easily scale up to networks of millions of nodes while significantly improves over

previous state-of-the-arts in terms of the accuracy of the estimated influence and the quality of the selected nodes in maximizing the influence.

## 4.1 Introduction

Motivated by applications in viral marketing [86], researchers have been studying the influence maximization problem: find a set of nodes whose initial adoptions of certain idea or product can trigger, *in a time window*, the largest expected number of follow-ups. For this purpose, it is essential to accurately and efficiently estimate the number of follow-ups of an arbitrary set of source nodes within the given time window. This is a challenging problem for that we need first accurately model the timing information in cascade data and then design a scalable algorithm to deal with large real-world networks.

Most existing studies for influence estimation and maximization in literature [100, 28, 29, 60] are based on the discrete-time diffusion models that first appeared in [138, 86]. However, based on our previous discussions and a sequence of most recent work [44, 41, 56, 59, 175, 176], we argue that modeling the information diffusion with the continuous-time model can achieve significantly better performance. There is a twofold rationale behind this modeling choice. First, since follow-ups occur asynchronously, continuous variables seem more appropriate to represent them. Artificially discretizing the time axis into bins introduces additional tuning parameters, like the bin size, which are not easy to choose optimally. Second, discrete time models can only describe transmission times which obey an exponential density, and hence can be too restricted to capture the rich temporal dynamics in the data. Extensive experimental comparisons on both synthetic and real world data showed that continuous-time models yield significant improvement in settings such as recovering hidden diffusion network structures from cascade data [44, 56] and predicting the timings of future events [41].

However, estimating and maximizing influence based on continuous-time diffusion models also entail many challenges. First, the influence estimation problem in this setting is a difficult graphical model inference problem, *i.e.*, computing the marginal density of continuous variables in loopy graphical models. The exact answer can be computed only for very special cases. For example, Gomez-Rodriguez et al. [139] have shown that the problem can be solved exactly when the transmission functions are exponential densities, by using continuous time Markov processes theory. However, the computational complexity of such approach, in general, scales exponentially with the size and density of the network. Moreover, extending the approach to deal with arbitrary transmission functions would require additional nontrivial approximations which would increase even more the computational complexity. Second, it is unclear how to scale up influence estimation and maximization algorithms based on continuous-time diffusion models to millions of nodes. Especially in the maximization case, even a naive sampling algorithm for approximate inference is not scalable: $n$ sampling rounds need to be carried out for each node to estimate the influence, which results in an overall $O(n|\mathcal{V}||\mathcal{E}|)$ algorithm.

Thus, in this chapter, we describe a scalable algorithm which is able to perform influence estimation and maximization in networks consisting of millions of nodes with provable performance guarantees. The remainder of the chapter is organized as follows: in Section 4.2, we review the continuous-time independent cascade model from the perspective of probabilistic graphical models. In Section 4.3, we give the influence estimation formulation in continuous-time diffusion networks. In Section 4.4, we develop a scalable randomized algorithm to solve the influence estimation problem efficiently. In Section 4.5, we solve the influence maximization problem based on the proposed efficient influence estimation algorithm. Finally, we conduct comprehensive experimental evaluations on both synthetic and real-world datasets in Section 4.6 and summarize our contributions in Section 4.7.

Table 4.1: Table of symbols

| Symbol | Description |
|---|---|
| $\mathcal{G}$ | Directed network with node set $\mathcal{V}$ and edge set $\mathcal{E}$ |
| $f_{ji}(t)$ | Density of the diffusion time from node $j$ to $i$ |
| $\mathcal{Q}_i$ | Set of shortest paths from sources to node $i$ |
| $\mathcal{A}$ | Source node set |
| $\sigma(\mathcal{A}, T)$ | Influence of source set $\mathcal{A}$ by time $T$ |
| $C$ | Limit on the source set size $\lvert\mathcal{A}\rvert \leq C$ |
| $\mathcal{N}(s, T)$ | Neighborhood of node $s$ before time $T$ |
| $\mathcal{N}(\mathcal{A}, T)$ | Neighborhood of source node set $\mathcal{A}$ before time $T$ |
| $r_i$ | Unite rate exponential random variable associated with node $i$ |
| $\pi_i$ | Set of parents for node $i$ |
| $m$ | Number of random samples |

## 4.2 A Graphical Model Perspective

The continuous-time independent cascade model is essentially a directed graphical model for a set of *dependent* random variables, the infection times $t_i$ of the nodes, where the conditional independence structure is supported on the contact network $\mathcal{G}$ (see Appendix A.2 for more details). More formally, the joint density of $\{t_i\}_{i\in\mathcal{V}}$ can be expressed as

$$p\left(\{t_i\}_{i\in\mathcal{V}}\right) = \prod\nolimits_{i\in\mathcal{V}} p\left(t_i|\{t_j\}_{j\in\pi_i}\right), \tag{4.1}$$

where $\pi_i$ denotes the set of parents of node $i$ in a cascade-induced DAG, and $p(t_i|\{t_j\}_{j\in\pi_i})$ is the conditional density of infection $t_i$ at node $i$ given the infection times of its parents.

Instead of directly modeling the infection times $t_i$, we can focus on the set of mutually *independent* random transmission times $\tau_{ji} = t_i - t_j$. Interestingly, by switching from a node-centric view to an edge-centric view, we obtain a fully factorized joint density of the set of transmission times

$$p\left(\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}\right) = \prod\nolimits_{(j,i)\in\mathcal{E}} f_{ji}(\tau_{ji}), \tag{4.2}$$

Based on the Shortest-Path property of the independent cascade model, each variable $t_i$ can be viewed as a transformation from the collection of variables $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$.

More specifically, let $\mathcal{Q}_i$ be the collection of directed paths in $\mathcal{G}$ from the source nodes to node $i$, where each path $q \in \mathcal{Q}_i$ contains a sequence of directed edges $(j, l)$. Assuming all source nodes are infected at zero time, then we obtain variable $t_i$ via

$$t_i = g_i\left(\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}\right) := \min_{q\in\mathcal{Q}_i} \sum\nolimits_{(j,l)\in q} \tau_{jl}, \tag{4.3}$$

where the transformation $g_i(\cdot)$ is the value of the shortest-path minimization. As a special case, we can now compute the probability of node $i$ infected before $T$ using a set of independent variables:

$$\Pr\{t_i \leq T\} = \Pr\left\{g_i\left(\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}\right) \leq T\right\}. \tag{4.4}$$

The significance of the relation is that it allows us to transform a problem involving a sequence of dependent variables $\{t_i\}_{i\in\mathcal{V}}$ to one with independent variables $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$. Furthermore, the two perspectives are connected via the shortest path algorithm in weighted directed graph, a standard well-studied operation in graph analysis.

## 4.3 Influence Estimation Problem in Continuous-Time Diffusion Networks

Intuitively, given a time window, the wider the spread of infection, the more influential the set of sources. We adopt the definition of influence as the average number of infected nodes given a set of source nodes and a time window, as in previous work [139]. More formally, consider a set of $C$ source nodes $\mathcal{A} \subseteq \mathcal{V}$ which gets infected at time zero, then, given a time window $T$, a node $i$ is infected in the time window if $t_i \leq T$. The expected number of infected nodes (or the influence) given the set of transmission functions $\{f_{ji}\}_{(j,i)\in\mathcal{E}}$ can be computed as

$$\sigma(\mathcal{A}, T) = \mathbb{E}\left[\sum\nolimits_{i\in\mathcal{V}} \mathbb{I}\{t_i \leq T\} | \mathcal{A}\right] = \sum\nolimits_{i\in\mathcal{V}} \Pr\{t_i \leq T | \mathcal{A}\}, \tag{4.5}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function, and the expectation is taken over the the set of *dependent* variables $\{t_i\}_{i\in\mathcal{V}}$.

Essentially, the influence estimation problem in Equation (4.5) is an inference problem for graphical models, where the probability of event $t_i \leq T$ given sources in $\mathcal{A}$ can be obtained by summing out the possible configuration of other variables $\{t_j\}_{j\neq i}$. That is

$$\Pr\{t_i \leq T|\mathcal{A}\} = \int_0^\infty \cdots \int_{t_i=0}^{T} \cdots \int_0^\infty \left(\prod\nolimits_{j\in\mathcal{V}} p\left(t_j|\{t_l\}_{l\in\pi_j}\right)\right)\left(\prod\nolimits_{j\in\mathcal{V}} dt_j\right), \quad (4.6)$$

which is, in general, a very challenging problem. First, the corresponding directed graphical models can contain nodes with high in-degree and high out-degree. For example, in Twitter, a user can follow dozens of other users, and another user can have hundreds of "followees". The tree-width corresponding to this directed graphical model can be very high, and we need to perform integration for functions involving many continuous variables. Second, the integral in general can not be evaluated analytically for heterogeneous transmission functions, which means that we need to resort to numerical integration by discretizing the domain $[0, \infty)$. If we use $N$ level of discretization for each variable, we would need to enumerate $O(N^{|\pi_i|})$ entries, exponential in the number of parents.

Only in very special cases, can one derive the closed-form equation for computing $\Pr\{t_i \leq T|\mathcal{A}\}$ [139]. However, without further heuristic approximation, the computational complexity of the algorithm is exponential in the size and density of the network. The intrinsic complexity of the problem entails the utilization of approximation algorithms, such as mean field algorithms or message passing algorithms. We will design an efficient randomized (or sampling) algorithm in the next section.

## 4.4 Efficient Influence Estimation in Continuous-Time Diffusion Networks

Our first key observation is that we can transform the influence estimation problem in Equation (4.5) into a problem with *independent* variables. Using the elegant relation in Equation (4.4), we have

$$\begin{aligned}\sigma(\mathcal{A}, T) &= \sum\nolimits_{i\in\mathcal{V}} \Pr\left\{g_i\left(\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}\right) \leq T|\mathcal{A}\right\} \\ &= \mathbb{E}\left[\sum\nolimits_{i\in\mathcal{V}} \mathbb{I}\left\{g_i\left(\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}\right) \leq T\right\}|\mathcal{A}\right], \end{aligned} \tag{4.7}$$

where the expectation is with respect to the set of independent variables $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$. This equivalent formulation suggests a naive sampling (NS) algorithm for approximating $\sigma(\mathcal{A}, T)$: draw $n$ samples of $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$, run a shortest path algorithm for each sample, and finally average the results (see Appendix A.3 for more details). However, this naive sampling approach has a computational complexity of $O(nC|\mathcal{V}||\mathcal{E}| + nC|\mathcal{V}|^2 \log|\mathcal{V}|)$ due to the repeated calling of the shortest path algorithm. This is quadratic to the network size, and hence not scalable to millions of nodes.

Our second key observation is that for each sample $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$, we are only interested in the neighborhood size of the source nodes, *i.e.*, the summation $\sum_{i\in\mathcal{V}} \mathbb{I}\{\cdot\}$ in Equation (4.7), rather than in the individual shortest paths. Fortunately, the neighborhood size estimation problem has been studied in the theoretical computer science literature. Here, we adapt a very efficient randomized algorithm by Cohen [32] to our influence estimation problem. This randomized algorithm has a computational complexity of $O(|\mathcal{E}| \log|\mathcal{V}| + |\mathcal{V}| \log^2|\mathcal{V}|)$ and it estimates the neighborhood sizes for *all* possible single source node locations. Since it needs to run once for each sample of $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$, we obtain an overall influence estimation algorithm with $O(n|\mathcal{E}| \log|\mathcal{V}| + n|\mathcal{V}| \log^2|\mathcal{V}|)$ computation, nearly linear in network size.

### 4.4.1 Single-Source Neighborhood Size Estimation

Given a fixed set of edge transmission times $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$ and a source node $s$, infected at time 0, the neighborhood $\mathcal{N}(s,T)$ of a source node $s$ given a time window $T$ is the set of nodes within the distance $T$ from $s$, which is defined as the following:

$$\mathcal{N}(s,T) = \left\{ i \mid g_i\left(\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}\right) \leq T,\ i \in \mathcal{V} \right\}. \tag{4.8}$$

Instead of estimating $\mathcal{N}(s,T)$ directly, the algorithm will assign an exponentially distributed random label $r_i$ to each network node $i$. Then, it makes use of the fact that the minimum of a set of exponential random variables $\{r_i\}_{i\in\mathcal{N}(s,T)}$ will also be a exponential random variable, but with its parameter equals to the number of variables. That is if each $r_i \sim \exp(-r_i)$, then the smallest label within distance $T$ from source $s$, $r_* := \min_{i\in\mathcal{N}(s,T)} r_i$, will distribute as $r_* \sim \exp\{-|\mathcal{N}(s,T)|r_*\}$. Suppose we randomize over the labeling $m$ times, and obtain $m$ such least labels, $\{r_*^u\}_{u=1}^m$. Then the neighborhood size can be estimated as

$$|\mathcal{N}(s,T)| \approx \frac{m-1}{\sum_{u=1}^m r_*^u}. \tag{4.9}$$

which is shown to be an unbiased estimator of $|\mathcal{N}(s,T)|$ [33]. This is an interesting relation since it allows us to transform the counting problem in (4.8) to a problem of finding the minimum random label $r_*$. The key question is whether we can compute the least label $r_*$ efficiently, given random labels $\{r_i\}_{i\in\mathcal{V}}$ and any source node $s$.

Cohen [33] designed a modified Dijkstra's algorithm (Algorithm A.1) to construct a data structure $r_*(s)$, called least label list, for each node $s$ to support such query. Essentially, the algorithm starts with the node $i$ with the smallest label $r_i$, and then it traverses in breadth-first search fashion along the reverse direction of the graph edges to find all reachable nodes. For each reachable node $s$, the distance $d_*$ between $i$ and $s$, and $r_i$ are added to the end of $r_*(s)$. Then the algorithm moves to the node $i'$ with the second smallest label $r_{i'}$, and similarly find all reachable nodes. For each

reachable node $s$, the algorithm will compare the current distance $d_*$ between $i'$ and $s$ with the last recorded distance in $r_*(s)$. If the current distance is smaller, then the current $d_*$ and $r_{i'}$ are added to the end of $r_*(s)$. Then the algorithm move to the node with the third smallest label and so on. The algorithm is summarized in Algorithm A.1 in Appendix A.4.

Algorithm A.1 returns a list $r_*(s)$ per node $s \in \mathcal{V}$, which contains information about distance to the smallest reachable labels from $s$. In particular, each list contains pairs of distance and random labels, $(d, r)$, and these pairs are ordered as

$$\infty > d_{(1)} > d_{(2)} > \ldots > d_{(|r_*(s)|)} = 0 \tag{4.10}$$

$$r_{(1)} < r_{(2)} < \ldots < r_{(|r_*(s)|)}, \tag{4.11}$$

where $\{\cdot\}_{(l)}$ denotes the $l$-th element in the list. (see Appendix A.4 for an example). If we want to query the smallest reachable random label $r_*$ for a given source $s$ and a time $T$, we only need to perform a binary search on the list for node $s$:

$$r_* = r_{(l)}, \text{ where } d_{(l-1)} > T \geq d_{(l)}. \tag{4.12}$$

Finally, to estimate $|\mathcal{N}(s, T)|$, we generate $m$ *i.i.d.* collections of random labels, run Algorithm A.1 on each collection, and obtain $m$ values $\{r_*^u\}_{u=1}^m$, which we use on Equation (4.9) to estimate $|\mathcal{N}(i, T)|$. The computational complexity of Algorithm A.1 is $O(|\mathcal{E}| \log |\mathcal{V}| + |\mathcal{V}| \log^2 |\mathcal{V}|)$, with expected size of each $r_*(s)$ being $O(\log |\mathcal{V}|)$. Then the expected time for querying $r_*$ is $O(\log \log |\mathcal{V}|)$ using binary search. Since we need to generate $m$ set of random labels and run Algorithm A.1 $m$ times, the overall computational complexity for estimating the single-source neighborhood size for all $s \in \mathcal{V}$ is $O(m|\mathcal{E}| \log |\mathcal{V}| + m|\mathcal{V}| \log^2 |\mathcal{V}| + m|\mathcal{V}| \log \log |\mathcal{V}|)$. For large scale network, and when $m \ll \min\{|\mathcal{V}|, |\mathcal{E}|\}$, this randomized algorithm can be much more efficient than approaches based on directly calculating the shortest paths.

### 4.4.2 Multiple-Source Neighborhood Size Estimation

When we have a set of sources, $\mathcal{A}$, its neighborhood is the union of the neighborhoods of its constituent sources

$$\mathcal{N}(\mathcal{A}, T) = \bigcup\nolimits_{i \in \mathcal{A}} \mathcal{N}(i, T). \tag{4.13}$$

This is true because each source independently infects its downstream nodes. Furthermore, to calculate the least label list $r_*$ corresponding to $\mathcal{N}(\mathcal{A}, T)$, we can simply reuse the least label list $r_*(i)$ of each individual source $i \in \mathcal{A}$. More formally,

$$r_* = \min\nolimits_{i \in \mathcal{A}} \ \ \min\nolimits_{j \in \mathcal{N}(i,T)} r_j, \tag{4.14}$$

where the inner minimization can be carried out by querying $r_*(i)$. Similarly, after we obtain $m$ samples of $r_*$, we can estimate $|\mathcal{N}(\mathcal{A}, T)|$ using Equation (4.9). Importantly, very little additional work is needed when we want to calculate $r_*$ for a set of sources $\mathcal{A}$, and we can reuse work done for a single source. This is very different from a naive sampling approach where the sampling process needs to be done completely anew if we increase the source set. In contrast, using the randomized algorithm, only an additional constant-time minimization over $|\mathcal{A}|$ numbers is needed.

### 4.4.3 Overall Algorithm

So far, we have achieved efficient neighborhood size estimation of $|\mathcal{N}(\mathcal{A}, T)|$ with respect to a given set of transmission times $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$. Next, we will estimate the influence by averaging over multiple sets of samples for $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$. More specifically, the relation from (4.7)

$$\sigma(\mathcal{A}, T) = \mathbb{E}_{\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}} \left[|\mathcal{N}(\mathcal{A}, T)|\right] = \mathbb{E}_{\{\tau_{ji}\}} \mathbb{E}_{\{r^1,\ldots,r^m\}|\{\tau_{ji}\}} \left[\frac{m-1}{\sum_{u=1}^{m} r_*^u}\right], \tag{4.15}$$

suggests the following overall algorithm

Continuous-Time Influence Estimation (ConTinEst):

1. Sample $n$ sets of random transmission times $\{\tau_{ij}^l\}_{(j,i)\in\mathcal{E}} \sim \prod_{(j,i)\in\mathcal{E}} f_{ji}(\tau_{ji})$
2. Given a set of $\{\tau_{ij}^l\}_{(j,i)\in\mathcal{E}}$, sample $m$ sets of random labels $\{r_i^u\}_{i\in\mathcal{V}} \sim \prod_{i\in\mathcal{V}} \exp(-r_i)$
3. Estimate $\sigma(\mathcal{A},T)$ by sample averages $\sigma(\mathcal{A},T) \approx \frac{1}{n}\sum_{l=1}^{n}\left((m-1)/\sum_{u_l=1}^{m} r_*^{u_l}\right)$

Importantly, the number of random labels, $m$, does not need to be very large. Since the estimator for $|\mathcal{N}(A,T)|$ is unbiased [33], essentially the outer-loop of averaging over $n$ samples of random transmission times further reduces the variance of the estimator in a rate of $O(1/n)$. In practice, we can use a very small $m$ (*e.g.*, 5 or 10) and still achieve good results, which is also confirmed by our later experiments. Compared to [28], the novel application of Cohen's algorithm arises for estimating influence for multiple sources, which drastically reduces the computation by cleverly using the least-label list from single source. Moreover, we have the following theoretical guarantee (see Appendix A.5 for proof)

**Theorem 4.1** *Draw the following number of samples for the set of random transmission times*

$$n \geqslant \frac{C\Lambda}{\epsilon^2}\log\left(\frac{2|\mathcal{V}|}{\delta}\right) \tag{4.16}$$

*where* $\Lambda := \max_{\mathcal{A}:|\mathcal{A}|\leq C} 2\sigma(\mathcal{A},T)^2/(m-2) + 2Var(|\mathcal{N}(\mathcal{A},T)|)(m-1)/(m-2) + 2a\epsilon/3$ *and* $|\mathcal{N}(\mathcal{A},T)| \leq a$*, and for each set of random transmission times, draw* $m$ *set of random labels. Then* $|\widehat{\sigma}(\mathcal{A},T) - \sigma(\mathcal{A},T)| \leqslant \epsilon$ *uniformly for all* $\mathcal{A}$ *with* $|\mathcal{A}| \leqslant C$*, with probability at least* $1-\delta$*.*

The theorem indicates that the minimum number of samples, $n$, needed to achieve certain accuracy is related to the actual size of the influence $\sigma(\mathcal{A},T)$, and the variance of the neighborhood size $|\mathcal{N}(\mathcal{A},T)|$ over the random draw of samples. The number of random labels, $m$, drawn in the inner loop of the algorithm will monotonically decrease

the dependency of $n$ on $\sigma(\mathcal{A}, T)$. It suffices to draw a small number of random labels, as long as the value of $\sigma(\mathcal{A}, T)^2/(m-2)$ matches that of $Var(|\mathcal{N}(\mathcal{A}, T)|)$. Another implication is that influence at larger time window $T$ is harder to estimate, since $\sigma(\mathcal{A}, T)$ will generally be larger and hence require more random labels.

## 4.5 Continuous-Time Influence Maximization

Once we know how to estimate the influence $\sigma(\mathcal{A}, T)$ for any $\mathcal{A} \subseteq \mathcal{V}$ and time window $T$ efficiently, we can use them in finding the optimal set of $C$ source nodes $\mathcal{A}^* \subseteq \mathcal{V}$ such that the expected number of infected nodes in $\mathcal{G}$ is maximized at $T$. That is, we seek to solve,

$$\mathcal{A}^* = \text{argmax}_{|\mathcal{A}| \leqslant C} \ \sigma(\mathcal{A}, T), \tag{4.17}$$

where set $\mathcal{A}$ is the variable. The above optimization problem is NP-hard in general. By construction, $\sigma(\mathcal{A}, T)$ is a non-negative, monotonic nondecreasing function in the set of source nodes, and it can be shown that $\sigma(\mathcal{A}, T)$ satisfies a diminishing returns property called submodularity [139].

A well-known approximation algorithm to maximize monotonic submodular functions is the *greedy algorithm*. It adds nodes to the source node set $\mathcal{A}$ sequentially. In step $k$, it adds the node $i$ which maximizes the *marginal gain* $\sigma(\mathcal{A}_{k-1} \cup \{i\}; T) - \sigma(\mathcal{A}_{k-1}; T)$. The greedy algorithm finds a source node set which achieves at least a constant fraction $(1 - 1/e)$ of the optimal [121]. Moreover, lazy evaluation [100] can be employed to reduce the required number of *marginal gains* per iteration. By using our influence estimation algorithm in each iteration of the greedy algorithm, we gain the following additional benefits:

First, at each iteration $k$, we do not need to rerun the full influence estimation algorithm. We just need to store the least label list $r_*(i)$ for each node $i \in \mathcal{V}$ computed for a single source, which requires expected storage size of $O(|\mathcal{V}| \log |\mathcal{V}|)$ overall.

Second, our influence estimation algorithm can be easily parallelized. Its two nested sampling loops can be parallelized in a straightforward way since the variables are independent of each other. However, in practice, we use a small number of random labels, and $m \ll n$. Thus we only need to parallelize the sampling for the set of random transmission times $\{\tau_{ji}\}$. The storage of the least element lists can also be distributed.

However, by using our randomized algorithm for influence estimation, we also introduce a sampling error to the greedy algorithm due to the approximation of the influence $\sigma(\mathcal{A}, T)$. Fortunately, the greedy algorithm is tolerant to such sampling noise, and a well-known result provides a guarantee for this case (following an argument in [94, Th. 7.9]):

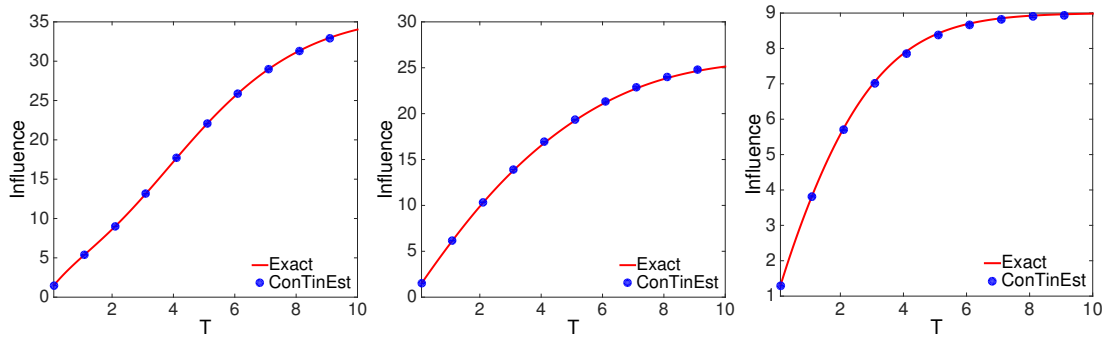
**Theorem 4.2** *Suppose the influence $\sigma(\mathcal{A}, T)$ for all $\mathcal{A}$ with $|\mathcal{A}| \leq C$ are estimated uniformly with error $\epsilon$ and confidence $1 - \delta$, the greedy algorithm returns a set of sources $\widehat{\mathcal{A}}$ such that $\sigma(\widehat{\mathcal{A}}, T) \geq (1 - 1/e)OPT - 2C\epsilon$ with probability at least $1 - \delta$.*

## 4.6 Experiments

We evaluate the accuracy of the estimated influence given by ConTinEst and investigate the performance of influence maximization on synthetic and real networks. We show that our approach significantly outperforms the state-of-the-art methods in terms of both speed and solution quality.

### 4.6.1 Synthetic Data

**Synthetic network generation.** We generate Kronecker networks [102]: (*i*) core-periphery networks (parameter matrix: [0.9 0.5; 0.5 0.3]), which mimic the information diffusion traces in real world networks, (*ii*) random networks ([0.5 0.5; 0.5 0.5]), typically used in physics and graph theory [47] and (*iii*) hierarchical networks ([0.9 0.1; 0.1 0.9]) [56]. Next, we assign a pairwise transmission function for every directed




(a) Core-periphery (b) Random (c) Hierarchical

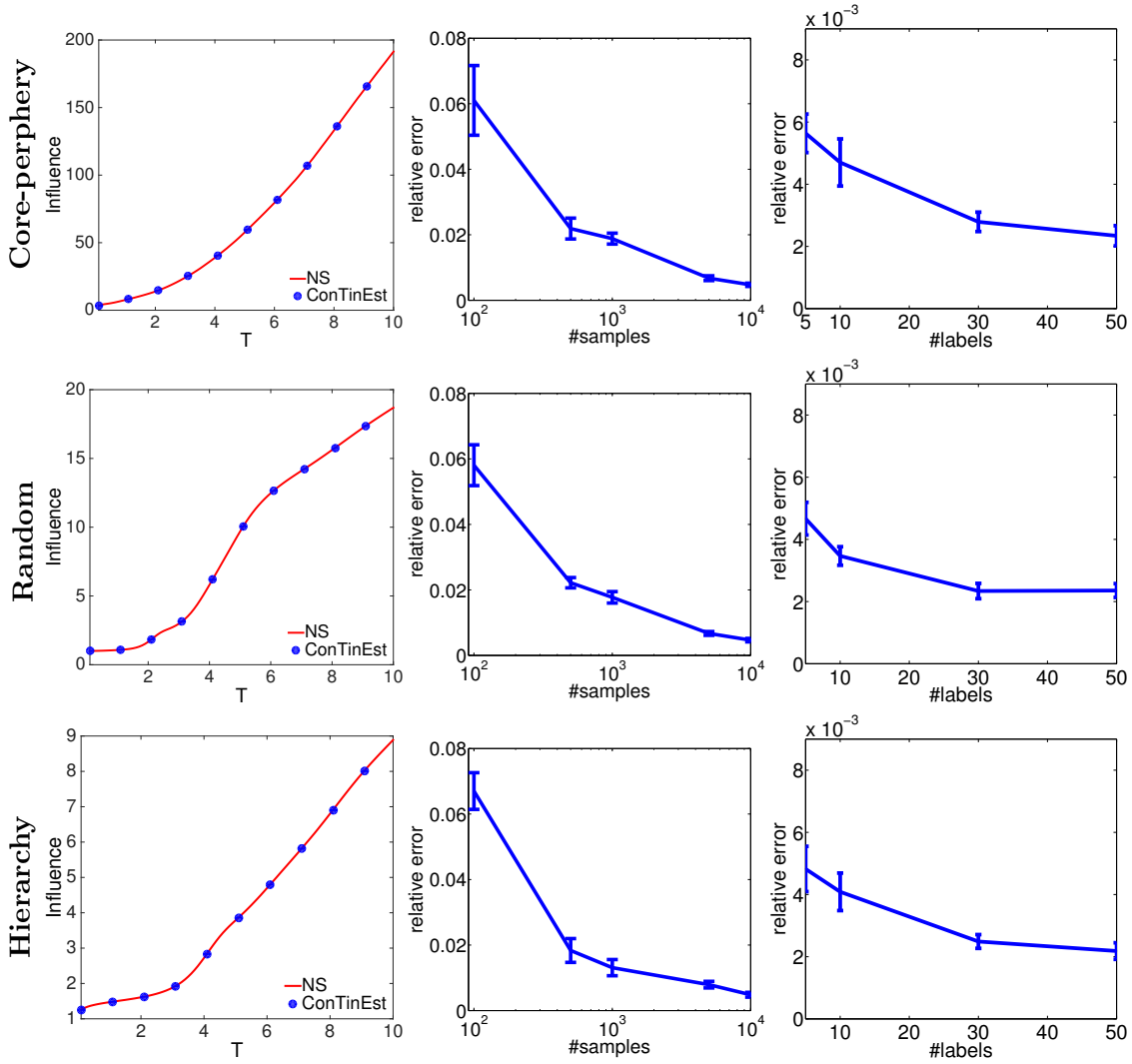
Figure 4.1: Estimated influence on three different types of networks with exponential edge transmission functions. Each type of network consists of 128 nodes and 141 edges. For ConTinEst, we draw 10,000 random samples, each of which has 5 random labels for each node.

edge in each type of network and set its parameters at random. In our experiments, we use the following Weibull distribution [97, 1],

$$f(t;\alpha,\beta) = \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1} e^{-(t/\alpha)^{\beta}}, t \geqslant 0, \tag{4.18}$$

where $\alpha > 0$ is a scale parameter and $\beta > 0$ is a shape parameter. The Weibull distribution (Wbl) has often been used to model lifetime events in survival analysis, providing more flexibility than an exponential distribution [97, 1]. We choose $\alpha$ and $\beta$ from $(0, 10)$ uniformly at random for each edge in order to have heterogeneous temporal dynamics. Finally, for each type of Kronecker network, we generate 10 sample networks, each of which has different $\alpha$ and $\beta$ chosen for every edge.

**Accuracy of the estimated influence.** First, we compare our influence estimation algorithm on several sparse small networks with exponential transmission functions where the exact solutions can be computed analytically by [139]. We set the parameter of every exponential transmission function by drawing a sample from a uniform distribution from (0, 10). We summarize the results in Figure 4.1, where we choose the highest degree node in the networks as the source and, for ConTinEst, we draw 10,000 random samples, each of which has 5 random labels for each node.



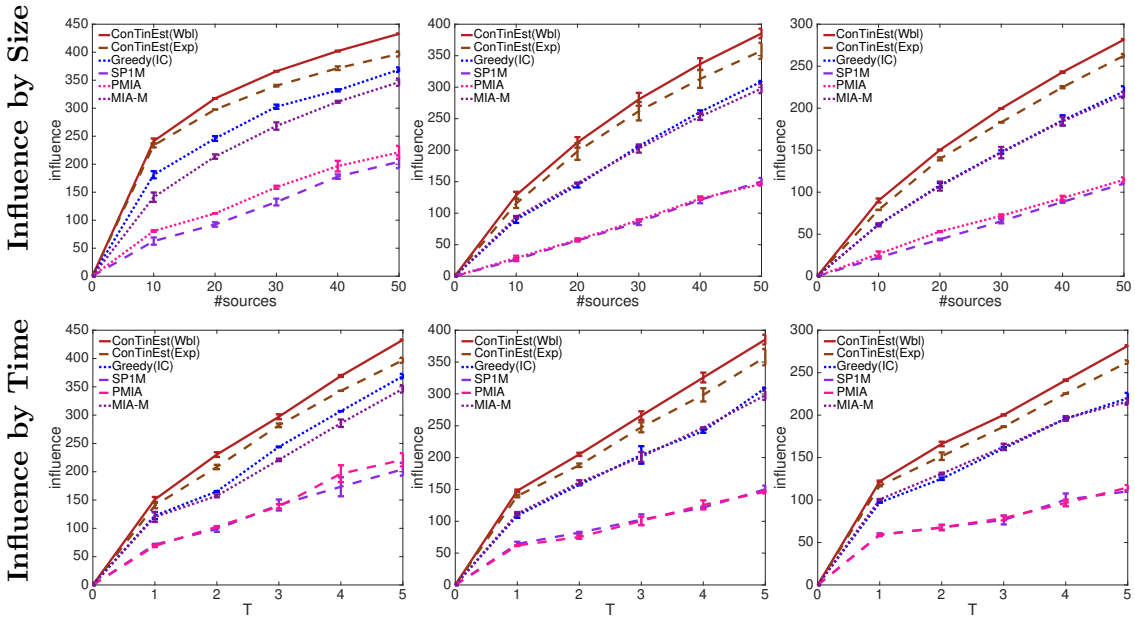

(a) Influence vs. time (b) Influence vs. #samples (c) Error vs. #labels

Figure 4.2: Influence estimation for core-periphery, random, and hierarchical networks with 1,024 nodes and 2,048 edges. Column (a) shows estimated influence by NS (near ground truth), and ConTinEst for increasing time window $T$; Column (b) shows ConTinEst's relative error against number of samples with 5 random labels and $T = 10$; Column (c) reports ConTinEst's relative error against number of random labels with 10,000 random samples and $T = 10$.

Remarkably, ConTinEst outputs values of influence which are very close to the (exact) values given by the exact influence estimation algorithm, with a relative error less than 0.01 in all three types of networks.

Then, we further compare ConTinEst to the Naive Sampling (NS) approach (see Appendix A.3) on several large networks with Weibull transmission functions, where to the best of our knowledge, there is no analytical solution to the influence estimation given Weibull transmission function. We take the highest degree node in a network as the source, and draw 1,000,000 samples for NS to obtain near ground truth. In Figures 4.2, Column (a) compares ConTinEst with the ground truth provided by NS at different time window $T$, from 0.1 to 10 in networks of different structures. For ConTinEst, we generate up to 10,000 random samples (or set of random waiting times), and 5 random labels in the inner loop. In all three networks, estimation provided by ConTinEst fits accurately the ground truth, and the relative error decreases quickly as we increase the number of samples and labels (Column (b) and Column (c)). For 10,000 random samples with 5 random labels, the relative error is smaller than 0.01.

**Influence Maximization** We compare ConTinEst to other influence maximization methods based on discrete-time diffusion models: traditional greedy [86], with discrete-time Linear Threshold Model (LT) and Independent Cascade Model (IC) diffusion models, and the heuristic methods SP1M [28], PMIA [27] and MIA-M [26]. For Influmax, since it only supports exponential pairwise transmission functions, we fit an exponential distribution per edge. Furthermore, Influmax is not scalable; when the average network density of the synthetic networks is $\sim 2.0$, the run time for Influmax is more than 24 hours. Instead, we present the results of ConTinEst using fitted exponential distributions (Exp).




(a) Core-periphery (b) Random (c) Hierarchy

Figure 4.3: Influence $\sigma(\mathcal{A}; T)$ achieved by varying number of sources $|\mathcal{A}|$ and observation window $T$ on the networks of different structures with 1,024 nodes, 2,048 edges and heterogeneous Weibull transmission functions. Top Row: influence against #sources by $T = 5$; Bottom Row: influence against the time window $T$ using 50 sources.

For the discrete-time IC model, we learn the infection probability within time window $T$ using Netrapalli's method [122]. The learned pairwise infection probabilities are also served for SP1M, PMIA and MIA-M, which essentially approximately calculate the influence based on the IC model. The meeting probabilities of MIA-M are set by following the original work [26] where we assign each edge $(u, v)$ the meeting probability $c/(d^{out}(u) + c)$, $d^{out}$ is the out-degree of node $u$ and $c = 5$.

For the discrete-time LT model, we set the weight of each incoming edge to a node $u$ to the inverse of its in-degree, as in previous work [86], and choose each node's threshold uniformly at random.

The top row of Figure 4.3 compares the expected number of infected nodes against source set size for different methods. ConTinEst outperforms the rest, and the competitive advantage becomes more dramatic the larger the source set grows. The

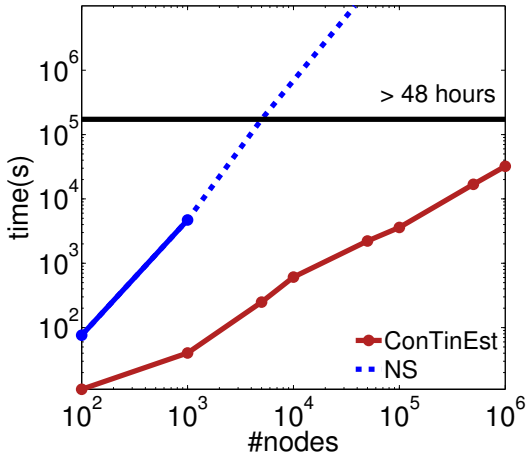
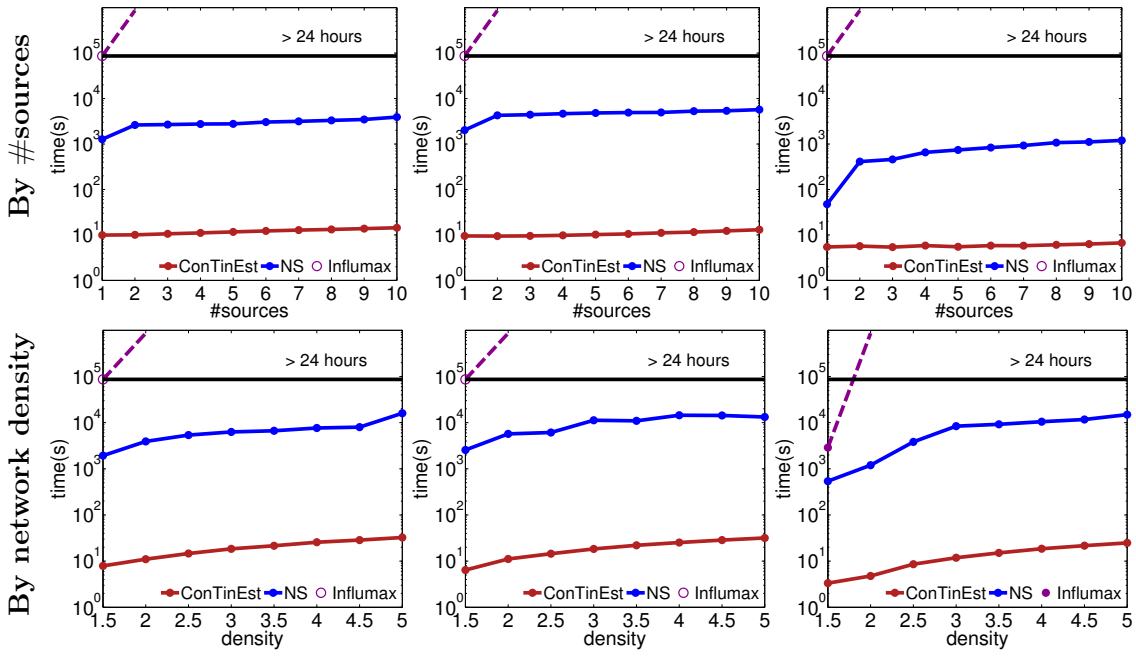



Figure 4.4: Scalability.

bottom row of Figure 4.3 shows the expected number of infected nodes against the time window for 50 selected sources. Again, ConTinEst performs the best for all three types of networks.

**Scalability.** We compare ConTinEst to Influmax [139] and the Naive Sampling (NS) method in terms of runtime for the continuous-time influence estimation and maximization. For ConTinEst, we draw 10,000 samples in the outer loop, each having 5 random labels in the inner loop. For NS, we also draw 10,000 samples. The first two experiments are carried out in a single 2.4GHz processor. First, we compare the performance of increasingly selecting sources (from 1 to 10) on small networks in the top row of Figure 4.5. When the number of selected sources is 1, different algorithms essentially spend time estimating the influence for each node. ConTinEst outperforms other methods by order of magnitude and for the number of sources larger than 1, it can efficiently reuse computations for estimating influence for individual nodes. Dashed lines mean that a method did not finish in 24 hours, and the estimated run time is plotted. Next, we compare the run time for selecting 10 sources on networks of 128 nodes with increasing densities (or the number of edges) in the bottom row of Figure 4.5. Again, Influmax and NS are order of magnitude slower due to their respective exponential and quadratic computational complexity



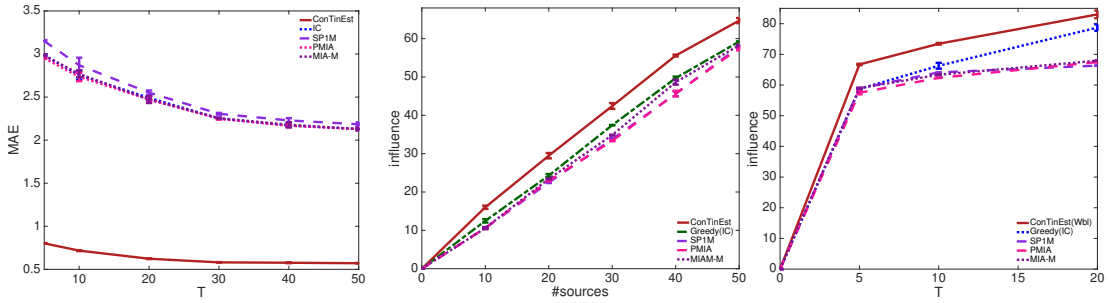

(a) Core-periphery (b) Random (c) Hierarchy

Figure 4.5: Running time comparison for the continuous-time influence maximization between ConTinEst and Influmax with $T = 10$. Top Row: runtime of selecting increasing number of sources in networks of 128 nodes and 320 edges; Bottom Row: selecting 10 sources in networks of 128 nodes with increasing density.

in network density. In contrast, the run time of ConTinEst only increases slightly with the increasing density since its computational complexity is linear in the number of edges. Finally, we evaluate the speed on large core-periphery networks, ranging from 100 to 1,000,000 nodes with density 1.5 in Figure 4.4. We report the parallel run time only for ConTinEst and NS (both are implemented by MPI running on 192 cores of 2.4Ghz) since Influmax is not scalable. In contrast to NS, the performance of ConTinEst increases linearly with the network size and can easily scale up to one million nodes.

### 4.6.2 Real Data

We first quantify how well each method can estimate the true influence in a real-world dataset. Then, we evaluate the solution quality of the selected sources for influence




(a) Influence estimation error (b) Influence vs. #sources (c) Influence vs. time

Figure 4.6: In MemeTracker dataset, (a) comparison of the accuracy of the estimated influence in terms of mean absolute error, (b) comparison of the influence of the selected nodes by fixing the observation window $T = 5$ and varying the number sources, and (c) comparison of the influence of the selected nodes by by fixing the number of sources to 50 and varying the time window.

maximization. We use the MemeTracker dataset [99] which has 10,967 hyperlink cascades among 600 media sites. We repeatedly split all cascades into a 80% training set and a 20% test set at random for five times. On each training set, we learn the continuous-time diffusion model assuming pairwise exponential transmission functions [56]. For discrete-time models, we learn the infection probabilities using [122] for IC, SP1M, PMIA and MIA-M.

**Comparing with real influence.** Let $\mathcal{C}(u)$ be the set of all cascades with $u$ being the source. Then, based on $\mathcal{C}(u)$, the total number of distinct nodes infected before $T$ quantifies the real influence of node $u$ up to time $T$. In Figure 4.6(a), we report the Mean Absolute Error (MAE) between the real and the estimated influence. Clearly, ConTinEst performs the best statistically. Because the length of real cascades empirically conforms to a power-law distribution where most cascades are very short (2-4 nodes), the gap of the estimation error is relatively not large. However, we emphasize that such accuracy improvement is critical for maximizing long-term influence. The estimation error will accumulate along the spreading paths. Hence, any consistent improvement in influence estimation can lead to significant improvement to the overall influence estimation and maximization task.

**Influence Maximization.** Since the estimation of the real influence of each source node is the foundation for the influence maximization problem, we also expect ConTinEst to find a set of sources with higher influence. Figures 4.6(b) and 4.6(c) confirm this intuition. We evaluate the influence of the selected nodes in the same spirit as influence estimation: the true influence is calculated as the total number of distinct nodes infected before $T$ based on $\mathcal{C}(u)$ of the selected nodes. The selected sources given by ConTinEst achieve the best performance as we change the number of selected sources and the observation time window.

## 4.7 Summary

In this chapter, we have developed an efficient randomized algorithm ConTinEst to make the inference about the expected influence of any given set of nodes based on the learned continuous-time diffusion models. This new algorithm is able to scale up linearly to networks of millions of nodes while maintaining superb predictive accuracy. We conduct theoretical analysis about the sample complexity of the algorithm showing that ConTinEst can achieve the accuracy of $\epsilon$ for influence estimation in a network with $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges using $O(1/\epsilon^2)$ samples and up to logarithmic factors $O(n|\mathcal{E}|+n|\mathcal{V}|)$ computations. When used as a subroutine in a greedy influence maximization approach, our algorithm is able to find a set of source nodes with provable performance guarantees. More remarkably, compared to the existing literatures where experimental evaluations are mainly conducted by simulations, we are the first to rigorously compare the predicted influence and evaluate the solution quality of the influence maximization task against independently-held true testing data, showing that the algorithm can select solutions which can indeed generate large-scale influence in real data. Finally, our work has also motivated a series of follow-ups [34, 155] to further fine-tune the performance of influence maximization in continuous-time diffusion networks.

# PART II RECURRENT PROCESSES

The multivariate terminating process captures one family of generating processes where at most one event can happen to each dimension. In many other cases, recurrent events can occur to the same dimension and between different pairs of entities. In the second part of the thesis, we illustrate the capability of our proposed framework in modeling the recurrence of high-dimensional event data. Specifically, in Chapter 5, we study the recurrent events occurring between different types of entities. For instance, users may repeatedly listen to the same album (or item) at different moments. Here, we mainly seek to propose stochastic processes that can accurately when the next event can happen between which pair of entities. Then, in Chapter 6, we consider the recurrent events that occur among entities of the same type. For example, the repeated user activities (posting, re-tweeting, commenting) in online social networks. The focus of this chapter is on estimating and controlling the overall expected user activities in large networks.

**Accurate Modeling:** we present the low-rank Hawkes process for capturing the recurrent actions between users and items to make time-sensitive recommendations and returning time predictions.

**Efficient Learning:** we develop a new optimization algorithm to learn the low rank Hawkes process model efficiently. Our algorithm blends proximal gradient and conditional gradient methods, and achieves the optimal $O(1/t)$ convergence rate.

**Scalable Inference:** based on the models for recurrent user activities, we derive a novel predictive formula for the overall network activity given the intensity of exogenous events of each individual user. We also propose a convex optimization framework to optimize the user engagement under certain budget constraints.

# CHAPTER V

# TIME-SENSITIVE RECOMMENDATION

Recurrent events are common and important phenomena in many real-life systems. For modern web companies, active engagement and high returning rates of users to the services are crucial for a healthy growth of business. In biomedical studies, many diseases and clinical outcomes might recur to the same patient. In some cases, recurrent events can happen between pairs of different types of entities (items). For instance, users might listen to different albums under specific contexts. People can develop various types of diseases during the process of getting aged. In other cases, users' activities can recur due to the excitation from other users. For example, online users might keep posting messages due to the comments and retweets from their friends. In this chapter, we are particularly interested in describing the modeling and learning capability of our framework to capture users' recurrent interactions with other types of items, like albums, products, diseases, etc. Then, in the next chapter, we focus on the inference capability of the framework to boost user activities, which has significant value of practice to modern social networking services.

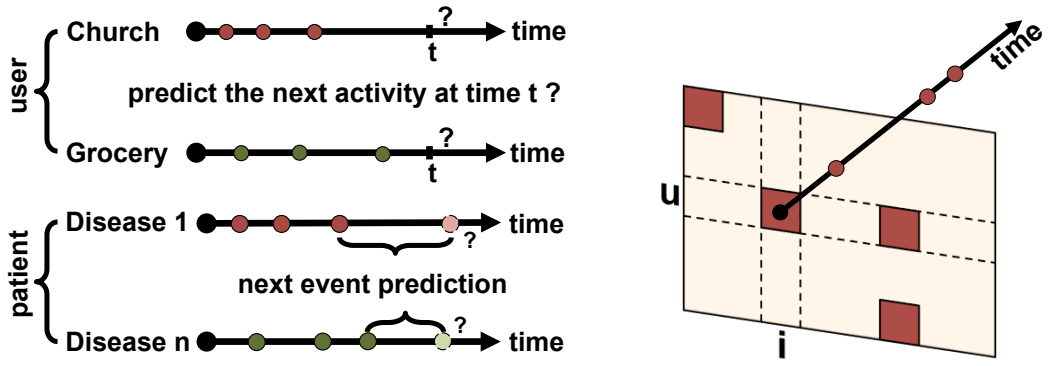
One fundamental goal of studying the interactions between users and items is to provide personalized services and user experiences. By making personalized suggestions, a recommender system is playing a crucial role in improving the engagement of users in modern web-services. However, most recommendation algorithms do not explicitly take into account the temporal behavior and the recurrent activities of users. Two central but less explored questions are how to recommend the most desirable item *at the right moment*, and how to predict *the next returning time* of a user to a service. To address these questions, we propose a novel low-rank multivariate point process

which connects self-exciting point processes and low-rank models to capture the recurrent temporal patterns in a large collection of user-item consumption pairs. We show that the parameters of the model can be estimated via a convex optimization, and furthermore, we develop an efficient algorithm that maintains $O(1/\epsilon)$ convergence rate, scales up to problems with millions of user-item pairs and hundreds of millions of temporal events. Compared to other state-of-the-arts in both synthetic and real datasets, our model achieves superb predictive performance in the two time-sensitive recommendation tasks. Finally, we point out that our formulation can incorporate other extra context information of users, such as profile, textual and spatial features.

## 5.1 Introduction

Delivering personalized user experiences is believed to play a crucial role in the long-term engagement of users to modern web-services [171]. For example, making recommendations on proper items at the right moment can make personal assistant services on mainstream mobile platforms more competitive and usable, since people tend to have different activities depending on the temporal/spatial contexts such as morning vs. evening, weekdays vs. weekend (see for example Figure 5.1(a)). Unfortunately, most existing recommendation techniques are mainly optimized at predicting users' one-time preference (often denoted by integer ratings) on items, while users' continuously time-varying preferences remain largely under explored.

Besides, traditional user feedback signals (*e.g.* user-item ratings, click-through-rates, *etc.*) have been increasingly argued to be ineffective to represent real engagement of users due to the sparseness and nosiness of the data [171]. The *temporal patterns* at which users return to the services (items) thus becomes a more relevant metric to evaluate their satisfactions [83]. Furthermore, successful predictions of the returning time not only allows a service to keep track of the evolving user preferences, but also helps a service provider to improve their marketing strategies. For most web




(a) Predictions from recurrent events. (b) User-item-event model.

Figure 5.1: Time-sensitive recommendation. (a) in the top figure, one wants to predict the most desirable activity at a given time $t$ for a user; in the bottom figure, one wants to predict the returning time to a particular disease of a patient. (b) The sequence of events induced from each user-item pair $(u, i)$ is modeled as a temporal point process along time.

companies, if we can predict when users will come back next, we could make ads bidding more economic, allowing marketers to bid on time slots. After all, marketers need not blindly bid all time slots indiscriminately. In the context of modern electronic health record data, patients may have several diseases that have complicated dependencies on each other shown at the bottom of Figure 5.1(a). The occurrence of one disease could trigger the progression of another. Predicting the returning time on certain disease can effectively help doctors to take proactive steps to reduce the potential risks. However, since most models in literature are particularly optimized for predicting ratings [93, 137, 89, 30, 168, 85, 128], exploring the recurrent temporal dynamics of users' returning behaviors over time becomes more imperative and meaningful than ever before.

Although the aforementioned applications come from different domains, we seek to capture them in a unified model by addressing the following two related questions: (1) *how to recommend the most relevant item at the right moment*, and (2) *how to accurately predict the next returning-time of users to existing services*. The very recent work of Kapoor et al. [83, 84] is most related to these problems. Kapoor et al. [83]

Table 5.1: Table of symbols

| Symbol | Description |
|---|---|
| $\lambda^{u,i}(t)$ | Conditional intensity function between user $u$ and item $i$ |
| $\gamma_0^{u,i}$ | Base intensity value between user $u$ and item $i$ |
| $t_j^{u,i}$ | The time for the $j$-th event happens between user $u$ and item $i$ |
| $m$ | Total number of users |
| $n$ | Total number of items |
| $\boldsymbol{\Lambda}_0$ | $m$-by-$n$ base intensity matrix |
| $\boldsymbol{A}$ | $m$-by-$n$ Self-exciting matrix |
| $\mathcal{T}^{u,i}$ | Event sequence between user $u$ and item $i$ |
| $\boldsymbol{X}_1$ | Concatenation of matrix $\boldsymbol{\Lambda}_0$ and $\boldsymbol{A}$ by column $[\boldsymbol{\Lambda}_0; \boldsymbol{A}]$ |
| $\boldsymbol{X}_2$ | Concatenation of matrix $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$ by column $[\boldsymbol{Z}_1; \boldsymbol{Z}_2]$ |
| $\boldsymbol{X}$ | Concatenation of matrix $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ by column $[\boldsymbol{X}_1; \boldsymbol{X}_2]$ |
| $\rho, \lambda, \beta$ | Regularization coefficients |
| $\delta^k$ | Step size at iteration $k$ |

propose to solve the returning time prediction problem for music streaming service based on survival analysis [1]. They later extend the model into a two-state hidden semi-markov model [84], so that conditioned on a particular state, they are able to recommend one of the *existing* items at a specific moment. Although these models explicitly take the temporal dynamics of user-item pairs into consideration, a significant limitation is that the models do not naturally generalize to recommend any new item future in time, which is a crucial difference compared to our approach. Moreover, because survival analysis is often suitable for modeling a single terminal event [1], such as infection and death, it assumes that the inter-event time to be independent. However, in many cases this assumption might not hold. Another sequence of related temporal models [93, 137, 89] seek to predicting future user-item ratings by using time decaying functions which weight recent ratings more than old ones, so that the predicted ratings become time-sensitive. Other alternatives [30, 168, 85, 132, 128] extend the user-item matrix into a three-way user-item-time tensor, and based on tensor factorization, they can predict the ratings at an given time slot. The main

disadvantage of these models lies in the facts that they cannot extrapolate beyond a predefined observation window, and their performance depends on how we partition the time into intervals, which is often hard to tune in practice. Moreover, such methods cannot make any predictions of users' future returning time. In contrast, our approach explicitly models the temporal events from each user-item pair as a point process which naturally captures the asynchronous nature of the incoming events.

The chapter is then organized as follows: in Section 5.2, we propose a novel convex formulation of the problems by establishing an under explored connection between self-exciting point processes and low-rank models. In Section 5.3, we develop a new optimization algorithm to solve the low rank Hawkes process estimation problem efficiently. Our algorithm blends proximal gradient and conditional gradient methods, and achieves the optimal $O(1/t)$ convergence rate. In Section 5.4, we describe how to use our model for time-sensitive recommendations. Section 5.5 verifies that the learning algorithm scales up to millions of user-item pairs and hundreds of millions of temporal events, and achieves superb predictive performance on the two time-sensitive problems on both synthetic and real datasets. Finally, we summarize our contributions in Section 5.6.

## 5.2 Low-Rank Hawkes Processes

Among the examples of multivariate point processes, there is a flexible family, multivariate Hawkes process [72, 106], which can model phenomena such as self- and mutual-excitation of events occurring over time. Specifically, the conditional intensity at dimension $n$ is

$$\lambda_j(t) = \lambda_j^0 + \sum_{k=1}^{D} \sum_{t_i^k < t} \kappa_{jk}(t, t_i^k), \tag{5.1}$$

which captures the interdependency among events over time and across dimensions shown in Figure 5.2. The base conditional intensity $\lambda_n^0 \geq 0$ captures the rate of observing an event in dimension $j$ without the influence of preceding events. The

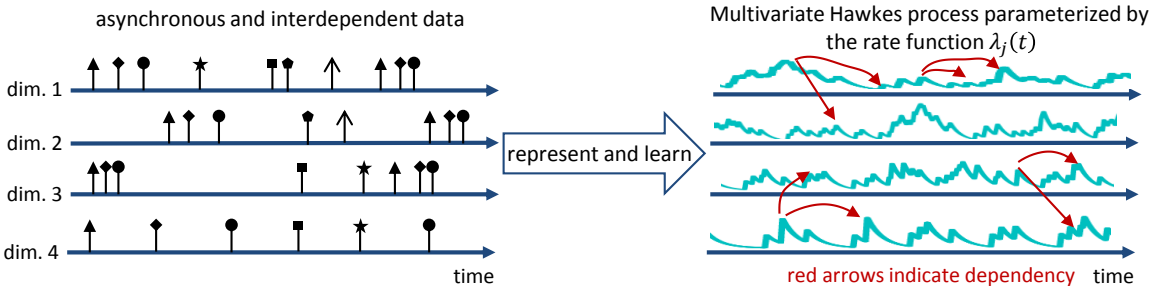



Figure 5.2: Multivariate Hawkes process for interdependent asynchronous and dependent event data.

function $\kappa_{jk}(t, t_i^k) \geq 0$ is a nonlinear *triggering* function of the past event timing $t_i^k$, and captures the influence of the particular past event on dimension $k$ at time $t_i^k$ to the occurrence of a new event on dimension $j$ at time $t$. One choice of the triggering function is an exponential kernel $\kappa_{jk}(t, t_i^k) = \alpha_{jk} e^{-\beta_{jk}(t-t_i^k)}$, which captures the notion that temporal dependency of events decays exponentially over time. Another example is a periodic basis function $\kappa_{jk}(t, t_i^k) = \alpha_{jk} \sin(2\pi(t - t_i^k)) + 1$, which captures the periodic dynamics of the event timing.

### 5.2.1 Modeling Recurrent User Activities with Hawkes Processes

Figure 5.1(b) highlights the basic setting of our model. For each observed user-item pair $(u, i)$, we model the occurrences of user $u$'s past consumption events on item $i$ as a self-exciting *Hawkes* process [72] with the intensity:

$$\lambda^{u,i}(t) = \lambda_0^{u,i} + \alpha^{u,i} \sum_{t_i \in \mathcal{T}} \gamma(t, t_i), \tag{5.2}$$

where $\gamma(t, t_i) \geqslant 0$ is the triggering kernel capturing temporal dependencies, $\alpha^{u,i} \geq 0$ scales the magnitude of the influence of each past event, $\lambda_0^{u,i} \geqslant 0$ is a baseline intensity, and the summation of the kernel terms is history dependent and thus a stochastic process by itself.

We have a twofold rationale behind this modeling choice. First, the baseline intensity $\lambda_0^{u,i}$ captures users' inherent and long-term preferences to items, regardless of the history. Second, the triggering kernel $\gamma(t, t_i)$ quantifies how the influence from

each past event evolves over time, which makes the intensity function depend on the history $\mathcal{T}$. Thus, a *Hawkes* process is essentially a conditional Poisson process [87] in the sense that conditioned on the history $\mathcal{T}$, the Hawkes process is a Poisson process formed by the superposition of a background homogeneous Poisson process with the intensity $\gamma_0$ and a set of inhomogeneous Poisson processes with the intensity $\gamma(t, t_i)$.

### 5.2.2 Transferring Knowledge with Low Rank Models

So far, we have shown modeling a sequence of events from a single user-item pair. Since we cannot observe the events from all user-item pairs, the next step is to transfer the learned knowledge to unobserved pairs. Given $m$ users and $n$ items, we represent the intensity function between user $u$ and item $i$ as $\lambda^{u,i}(t) = \lambda_0^{u,i} + \alpha^{u,i} \sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \gamma(t, t_j^{u,i})$, where $\lambda_0^{u,i}$ and $\alpha^{u,i}$ are the $(u,i)$-th entry of the $m$-by-$n$ non-negative base intensity matrix $\mathbf{\Lambda}_0$ and the self-exciting matrix $\boldsymbol{A}$, respectively.

However, the two matrices of coefficients $\mathbf{\Lambda}_0$ and $\boldsymbol{A}$ contain too many parameters. Since it is often believed that users' behaviors and items' attributes can be categorized into a limited number of prototypical types, we assume that $\mathbf{\Lambda}_0$ and $\boldsymbol{A}$ have low-rank structures. That is, the nuclear norms of these parameter matrices are small $\|\mathbf{\Lambda}_0\|_* \leqslant \lambda', \|\boldsymbol{A}\|_* \leqslant \beta'$. Some researchers also explicitly assume that the two matrices factorize into products of low rank factors. Here we assume the above nuclear norm constraints in order to obtain convex parameter estimation procedures later. In consequence, we define the low-rank Hawkes process as the following:

**Definition 5.1** *The Low-rank Hawkes Process is an $mn$-dimensional Hawkes process arranged in an $m$-by-$n$ grid. The conditional intensity function of the entry $(i,j)$ is $\lambda^{i,j}(t) = \mathbf{\Lambda}_0(i,j) + \boldsymbol{A}(i,j) \sum_{t_k^{i,j} < t} \gamma(t, t_k^{i,j})$ where $\mathbf{\Lambda}_0$ and $\boldsymbol{A}$ are the low-rank base intensity matrix and the self-exciting matrix, respectively.*

### 5.2.3 Triggering Kernel Parametrization and Extensions

Because it is only required that the triggering kernel should be nonnegative and bounded, the integral of the intensity function often has the analytic form when $\gamma(t, t_j^{u,i})$ belongs to many flexible parametric families, such as the Weibull and Log-logistic distributions [1]. For the simplest case, $\gamma(t, t_j^{u,i})$ takes the exponential form $\gamma(t, t_j^{u,i}) = \exp(-(t - t_j^{u,i})/\sigma)$. Alternatively, we can make the intensity function $\lambda^{u,i}(t)$ depend on other additional context information associated with each event. For instance, we can make the base intensity $\mathbf{\Lambda}_0$ depend on user-profiles and item-contents [41, 42]. We might also extend $\mathbf{\Lambda}_0$ and $\boldsymbol{A}$ into tensors to incorporate the additional spatial, textual, categorical, and user profile information. Furthermore, we can even learn the triggering kernel directly using nonparametric methods [44, 176]. Without loss of generality, we stick with the exponential form in later sections.

## 5.3 Efficient Learning Algorithm

Having presented our model, in this section, we develop a new algorithm which blends proximal gradient and conditional gradient methods to learn the model efficiently.

### 5.3.1 Convex Formulation

Let $\mathcal{T}^{u,i}$ be the set of events induced between $u$ and $i$. We express the log-likelihood of observing each sequence $\mathcal{T}^{u,i}$ based on (5.3) as:

$$\ell\left(\mathcal{T}^{u,i} | \mathbf{\Lambda}_0, \boldsymbol{A}\right) = \sum\nolimits_{t_j^{u,i} \in \mathcal{T}^{u,i}} \log(\boldsymbol{w}_{u,i}^\top \boldsymbol{\phi}_j^{u,i}) - \boldsymbol{w}_{u,i}^\top \boldsymbol{\psi}^{u,i}, \tag{5.3}$$

where $\boldsymbol{w}_{u,i} = (\mathbf{\Lambda}_0(u,i), \boldsymbol{A}(u,i))^\top$, $\boldsymbol{\phi}_j^{u,i} = (1, \sum_{t_k^{u,i} < t_j^{u,i}} \gamma(t_j^{u,i}, t_k^{u,i}))^\top$ , and $\boldsymbol{\psi}^{u,i} = (T, \sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \int_{t_j^{u,i}}^{T} \gamma(t, t_j^{u,i}) dt)^\top$. When $\gamma(t, t_j^{u,i})$ is the exponential kernel, $\boldsymbol{\psi}^{u,i}$ can be expressed as $\boldsymbol{\psi}^{u,i} = (T, \sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \sigma(1 - \exp(-(T - t_j^{u,i})/\sigma)))^\top$. Then, the log-likelihood of observing all event sequences $\mathcal{O} = \{\mathcal{T}^{u,i}\}_{u,i}$ is simply a summation of

each individual term by $\ell(\mathcal{O}) = \sum_{\mathcal{T}^{u,i} \in \mathcal{O}} \ell(\mathcal{T}^{u,i})$. Therefore, we can have the following optimization problem, which is convex in $\boldsymbol{\Lambda}_0$ and $\boldsymbol{A}$:

$$\text{OPT} = \min_{\boldsymbol{\Lambda}_0, \boldsymbol{A}} -\frac{1}{|\mathcal{O}|} \sum\nolimits_{\mathcal{T}^{u,i} \in \mathcal{O}} \ell\left(\mathcal{T}^{u,i} | \boldsymbol{\Lambda}_0, \boldsymbol{A}\right) + \lambda \|\boldsymbol{\Lambda}_0\|_* + \beta \|\boldsymbol{A}\|_* \\ \text{subject to } \boldsymbol{\Lambda}_0, \boldsymbol{A} \geq \mathbf{0}, \tag{5.4}$$

where the matrix nuclear norm $\|\cdot\|_*$, which is a summation of all singular values, is commonly used as a convex surrogate for the matrix rank function [142]. One off-the-shelf solution to 5.4 is proposed in [175] based on ADMM. However, the algorithm in [175] requires, at each iteration, a full SVD for computing the proximal operator, which is often prohibitive with large matrices. Alternatively, we might turn to more efficient conditional gradient algorithms [173], which require instead, the much cheaper linear minimization oracles. However, the non-negativity constraints in our problem prevent the linear minimization from having a simple analytical solution.

### 5.3.2 Alternative Convex Formulation

The difficulty of directly solving the original formulation 5.4 is caused by the fact that the nonnegative constraints are entangled with the non-smooth nuclear norm penalty. To address this challenge, we approximate 5.4 using a simple penalty method. Specifically, given $\rho > 0$, we arrive at the next formulation 5.5 by introducing two auxiliary variables $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$ with some penalty function, such as the squared Frobenius norm.

$$\widehat{\text{OPT}} = \min_{\boldsymbol{\Lambda}_0, \boldsymbol{A}, \boldsymbol{Z}_1, \boldsymbol{Z}_2} -\frac{1}{|\mathcal{O}|} \sum_{\mathcal{T}^{u,i} \in \mathcal{O}} \ell\left(\mathcal{T}^{u,i} | \boldsymbol{\Lambda}_0, \boldsymbol{A}\right) + \lambda \|\boldsymbol{Z}_1\|_* + \beta \|\boldsymbol{Z}_2\|_* + \rho \|\boldsymbol{\Lambda}_0 - \boldsymbol{Z}_1\|_F^2 \\ + \rho \|\boldsymbol{A} - \boldsymbol{Z}_2\|_F^2 \quad \text{subject to} \quad \boldsymbol{\Lambda}_0, \boldsymbol{A} \geq \mathbf{0}. \tag{5.5}$$

We show in Theorem 5.2 that when $\rho$ is properly chosen, these two formulations lead to the same optimum. See appendix for the complete proof. More importantly, the new formulation 5.5 allows us to handle the non-negativity constraints and nuclear norm regularization terms separately.

**Algorithm 5.1:** Learning Algorithm

**Input**: $\mathcal{O} = \{\mathcal{T}^{u,i}\}$, $\rho > 0$
**Output**: $\boldsymbol{Y}_1 = [\boldsymbol{\Lambda}_0; \boldsymbol{A}]$

Initialize $\boldsymbol{X}_1^0$ and $\boldsymbol{X}_2^0 = \boldsymbol{X}_1^0$ ;
Set $\boldsymbol{Y}^0 = \boldsymbol{X}^0$;
**for** $k = 1, 2, \ldots$ **do**
  $\delta^k = \frac{2}{k+1}$;
  $\boldsymbol{U}^{k-1} = (1-\delta^k)\boldsymbol{Y}^{k-1} + \delta^k \boldsymbol{X}^{k-1}$ ;
  $\boldsymbol{X}_1^k = \mathrm{Prox}_{\boldsymbol{U}^{k-1}}\left(\eta_k \nabla_1(f(\boldsymbol{U}^{k-1}))\right)$;
  $\boldsymbol{X}_2^k = \mathrm{LMO}_{\psi}\left(\nabla_2(f(\boldsymbol{U}^{k-1}))\right)$;
  $\boldsymbol{Y}^k = (1-\delta^k)\boldsymbol{Y}^{k-1} + \delta^k \boldsymbol{X}^k$;
**end**

**Algorithm 5.2:** $\mathrm{Prox}_{\boldsymbol{U}^{k-1}}\left(\eta_k \nabla_1(f(\boldsymbol{U}^{k-1}))\right)$

$\boldsymbol{X}_1^k = \left(\boldsymbol{U}^{k-1} - \eta_k \nabla_1(f(\boldsymbol{U}^{k-1}))\right)_+$;

**Algorithm 5.3:** $\mathrm{LMO}_{\psi}\left(\nabla_2(f(\boldsymbol{U}^{k-1}))\right)$

$(u_1, v_1), (u_2, v_2)$ top singular vector pairs of $-\nabla_2(f(\boldsymbol{U}^{k-1}))[\boldsymbol{Z}_1]$ and $-\nabla_2(f(\boldsymbol{U}^{k-1}))[\boldsymbol{Z}_2]$;
$\boldsymbol{X}_2^k[\boldsymbol{Z}_1] = u_1 v_1^\top$, $\boldsymbol{X}_2^k[\boldsymbol{Z}_2] = u_2 v_2^\top$;
Find $\alpha_1^k$ and $\alpha_2^k$ by solving (5.7);
$\boldsymbol{X}_2^k[\boldsymbol{Z}_1] = \alpha_1^k \boldsymbol{X}_2^k[\boldsymbol{Z}_1]$;
$\boldsymbol{X}_2^k[\boldsymbol{Z}_2] = \alpha_2^k \boldsymbol{X}_2^k[\boldsymbol{Z}_2]$;

**Theorem 5.2** *With the condition $\rho \geq \rho^*$, the optimal value $\widehat{\mathrm{OPT}}$ of the problem 5.5 coincides with the optimal value* OPT *in the problem (5.4) of interest, where $\rho^*$ is a problem dependent threshold,*

$$\rho^* = \max\left\{\frac{\lambda\left(\|\boldsymbol{\Lambda}_0^*\|_* - \|\boldsymbol{Z}_1^*\|_*\right) + \beta\left(\|\boldsymbol{A}^*\|_* - \|\boldsymbol{Z}_2^*\|_*\right)}{\|\boldsymbol{\Lambda}_0^* - \boldsymbol{Z}_1^*\|_F^2 + \|\boldsymbol{A}^* - \boldsymbol{Z}_2^*\|_F^2}\right\}.$$

### 5.3.3 Efficient Optimization: Proximal Method Meets Conditional Gradient

Now, we are ready to present Algorithm 5.1 for solving 5.5 efficiently. Denote $\boldsymbol{X}_1 = [\boldsymbol{\Lambda}_0; \boldsymbol{A}]$, $\boldsymbol{X}_2 = [\boldsymbol{Z}_1; \boldsymbol{Z}_2]$ and $\boldsymbol{X} = [\boldsymbol{X}_1; \boldsymbol{X}_2]$. We use the bracket $[\cdot]$ notation $\boldsymbol{X}_1[\boldsymbol{\Lambda}_0]$, $\boldsymbol{X}_1[\boldsymbol{A}]$, $\boldsymbol{X}_2[\boldsymbol{Z}_1]$, $\boldsymbol{X}_2[\boldsymbol{Z}_2]$ to represent the respective part for simplicity. Let $f(\boldsymbol{X}) := f(\boldsymbol{\Lambda}_0, \boldsymbol{A}, \boldsymbol{Z}_1, \boldsymbol{Z}_2) = -\frac{1}{|\mathcal{O}|}\sum_{\mathcal{T}^{u,i}\in\mathcal{O}} \ell\left(\mathcal{T}^{u,i}|\boldsymbol{\Lambda}_0, \boldsymbol{A}\right) + \rho\|\boldsymbol{\Lambda}_0 - \boldsymbol{Z}_1\|_F^2 + \rho\|\boldsymbol{A} - \boldsymbol{Z}_2\|_F^2$. The course of our action is straightforward: at each iteration, we apply cheap projection gradient for block $\boldsymbol{X}_1$ and cheap linear minimization for block $\boldsymbol{X}_2$ and maintain three interdependent sequences $\left\{\boldsymbol{U}^k\right\}_{k\geq 1}$, $\left\{\boldsymbol{Y}^k\right\}_{k\geq 1}$ and $\left\{\boldsymbol{X}^k\right\}_{k\geq 1}$ based on the accelerated scheme in [95, 96]. To be more specific, the algorithm consists of two main subroutines:

**Proximal Gradient.** When updating $\boldsymbol{X}_1$, we compute directly the associated proximal operator, which in our case, reduces to the simple projection $\boldsymbol{X}_1^k = \left(\boldsymbol{U}^{k-1} - \eta_k \nabla_1 f(\boldsymbol{U}^{k-1})\right)_+$, where $(\cdot)_+$ simply sets the negative coordinates to zero.

**Conditional Gradient.** When updating $\boldsymbol{X}_2$, instead of computing the proximal operator, we call the linear minimization oracle ($\text{LMO}_\psi$):

$$\boldsymbol{X}_2^k[\boldsymbol{Z}_1] = \operatorname{argmin} \left\{ \langle p_k[\boldsymbol{Z}_1], \boldsymbol{Z}_1 \rangle + \psi(\boldsymbol{Z}_1) \right\}, \tag{5.6}$$

where $p_k = \nabla_2(f(\boldsymbol{U}^{k-1}))$ is the partial derivative with respect to $\boldsymbol{X}_2$ and $\psi(\boldsymbol{Z}_1) = \lambda\|\boldsymbol{Z}_1\|_*$. We do similar updates for $\boldsymbol{X}_2^k[\boldsymbol{Z}_2]$. The overall performance clearly depends on the efficiency of this LMO, which can be solved efficiently in our case as illustrated in Algorithm 4. Following [172], the linear minimization for our situation requires only: **(i)** computing $\boldsymbol{X}_2^k[\boldsymbol{Z}_1] = \operatorname{argmin}_{\|\boldsymbol{Z}_1\|_* \leq 1} \langle p_k[\boldsymbol{Z}_1], \boldsymbol{Z}_1 \rangle$, where the minimizer is readily given by $\boldsymbol{X}_2^k[\boldsymbol{Z}_1] = u_1 v_1^\top$, and $u_1, v_1$ are the top singular vectors of $-p_k[\boldsymbol{Z}_1]$; and **(ii)** conducting a line-search that produces a scaling factor $\alpha_1^k = \operatorname{argmin}_{\alpha_1 \geq 0} h(\alpha_1)$

$$h(\alpha_1) := \rho\|\boldsymbol{Y}_1^{k-1}[\boldsymbol{\Lambda}_0] - (1-\delta^k)\boldsymbol{Y}_2^{k-1}[\boldsymbol{Z}_1] - \delta^k(\alpha_1 \boldsymbol{X}_2^k[\boldsymbol{Z}_1])\|_F^2 + \lambda\delta^k\alpha_1 + C, \tag{5.7}$$

where $C = \lambda(1-\delta^k)\|\boldsymbol{Y}_2^{k-1}[\boldsymbol{Z}_1]\|_*$. The quadratic problem (5.7) admits a closed-form solution and thus can be computed efficiently. We repeat the same process for updating $\alpha_2^k$ accordingly. Let $E = \sum_{\mathcal{T}^{u,i} \in \mathcal{O}} |\mathcal{T}^{u,i}|$ be the number of totally observed events. The computational complexity of Algorithm 5.1 is $O(E + 2mn + m^2)$.

### 5.3.4 Convergence Analysis

Denote $F(\boldsymbol{X}) = f(\boldsymbol{X}) + \psi(\boldsymbol{X}_2)$ as the objective in formulation 5.5, where $\boldsymbol{X} = [\boldsymbol{X}_1; \boldsymbol{X}_2]$. We establish the following convergence results for Algorithm 5.1 described above when solving formulation 5.5. Please refer to Appendix for complete proof.

**Theorem 5.3** *Let $\left\{\boldsymbol{Y}^k\right\}$ be the sequence generated by Algorithm 5.1 by setting $\delta^k = 2/(k+1)$, and $\eta^k = (\delta^k)^{-1}/L$. Then for $k \geq 1$, we have*

$$F(\boldsymbol{Y}^k) - \widehat{\text{OPT}} \leq \frac{4LD_1}{k(k+1)} + \frac{2LD_2}{k+1}. \tag{5.8}$$

*where $L$ corresponds to the Lipschitz constant of $\nabla f(\boldsymbol{X})$ and $D_1$ and $D_2$ are some problem dependent constants.*

**Remark.** Let $g(\Lambda_0, A)$ denote the objective in formulation 5.4, which is the original problem of our interest. By invoking Theorem 5.2, we further have, $g(\boldsymbol{Y}^k[\Lambda_0], \boldsymbol{Y}^k[A]) - \text{OPT} \leq \frac{4LD_1}{k(k+1)} + \frac{2LD_2}{k+1}$. The analysis builds upon the recursions from proximal gradient and conditional gradient methods. As a result, the overall convergence rate comes from two parts, as reflected in (5.8). Interestingly, one can easily see that for both the proximal and the conditional gradient parts, we achieve the respective *optimal* convergence rates. When there is no nuclear norm regularization term, the results recover the well-known optimal $O(1/t^2)$ rate achieved by proximal gradient method for smooth convex optimization. When there is no nonnegative constraint, the results recover the well-known $O(1/t)$ rate attained by conditional gradient method for smooth convex minimization. When both nuclear norm and non-negativity are in present, the proposed algorithm, up to our knowledge, is first of its kind, that achieves the best of both worlds, which could be of independent interest.

## 5.4 Inference

Once we have learned $\boldsymbol{\Lambda}_0$ and $\boldsymbol{A}$, we are ready to solve our proposed problems by using the following algorithms:

(a) **Item recommendation**. At any given time $t$, for each user-item pair $(u, i)$, because the intensity function $\lambda^{u,i}(t)$ indicates the tendency that user $u$ will consume item $i$ at time $t$, for each user $u$, we recommend the proper items by the following procedures:

1. Calculate $\lambda^{u,i}(t)$ for each item $i$.
2. Sort the items by the descending order of $\lambda^{u,i}(t)$.
3. Return the top-$k$ items.

(b) **Returning-time prediction**: for each user-item pair $(u, i)$, the intensity function $\lambda^{u,i}(t)$ dominates the point patterns along time. Given the history $\mathcal{T}^{u,i} =$

$\{t_1, t_2, \ldots, t_n\}$, we calculate the density of the next event time by $f(t|\mathcal{T}^{u,i}) = \lambda^{u,i}(t) \exp\left(-\int_{t_n}^{t} \lambda^{u,i}(t)dt\right)$, so we can use the expectation to predict the next event. Unfortunately, this expectation often does not have analytic forms due to the complexity of $\lambda^{u,i}(t)$ for Hawkes process, so we can use the numerical integration [130], or we can approximate the returning-time by:

1. Draw samples $\left\{t_{n+1}^1, \ldots, t_{n+1}^m\right\} \sim f(t|\mathcal{T}^{u,i})$ by Ogata's thinning algorithm D.1.
2. Estimate the returning-time by the sample average $\frac{1}{m}\sum_{i=1}^{m} t_{n+1}^i$

## 5.5 Experiments

We evaluate our algorithm by comparing with state-of-the-art competitors on both synthetic and real datasets. For each user, we randomly pick 20-percent of all the items she has consumed and hold out the *entire* sequence of events. Besides, for each sequence of the other 80-percent items, we further split it into a pair of training/testing subsequences. For each testing event, we evaluate the predictive accuracy on two tasks:

(a) **Item Recommendation**: suppose the testing event belongs to the user-item pair $(u, i)$. Ideally item $i$ should rank top at the testing moment. We record its predicted rank among all items. Smaller value indicates better performance.

(b) **Returning-Time Prediction:** we predict the returning-time from the learned intensity function and compute the absolute error with respect to the true time.

We repeat these two evaluations on all testing events. Because the predictive tasks on those entirely held-out sequences are much more challenging, we report the *total* mean absolute error (MAE) and that specific to the set of entirely *heldout* sequences, separately.

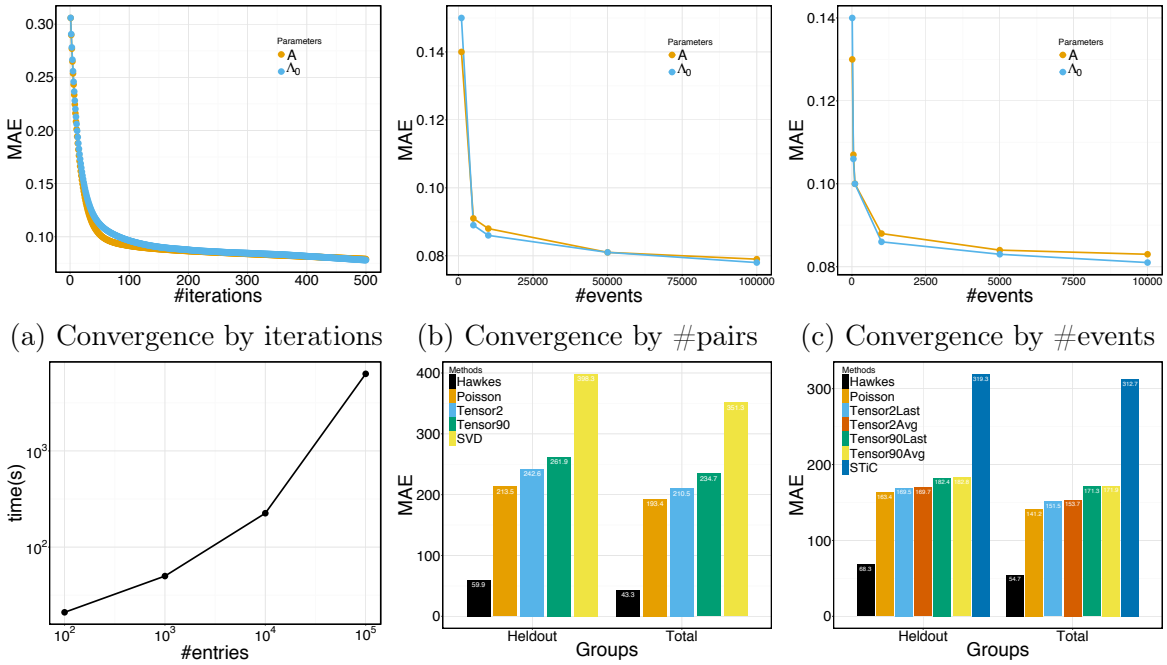
### 5.5.1 Baselines

**Poisson process** is a relaxation of our model by assuming each user-item pair $(u, i)$ has only a constant base intensity $\mathbf{\Lambda}_0(u, i)$, regardless of the history. For task (a), it gives static ranks regardless of the time. For task (b), it produces an estimate of the average inter-event gaps. In many cases, the Poisson process is a hard baseline in that the most popular items often have large base intensity, and recommending popular items is often a strong heuristic.

**STiC** [84] fits a semi-hidden Markov model to each observed user-item pair. Since it can only make recommendations specific to the few observed items visited before, instead of the large number of new items, we only evaluate its performance on the returning time prediction task. For the set of entirely held-out sequences, we use the average predicted inter-event time from each observed item as the final prediction.

**SVD** is the classic matrix factorization model. The implicit user feedback is converted into an explicit rating using the frequency of item consumption [15]. Since it is not designed for predicting the returning time, we report its performance on the time-sensitive recommendation task as a reference.

**Tensor factorization** generalizes matrix factorization to include time. We compare with the state-of-art method [30] which considers Poisson regression as the loss function to fit the number of events in each discretized time slot and shows better performance compared to other alternatives with the squared loss [168, 85, 128]. We report the performance by (1) using the parameters fitted only in the last interval, and (2) using the average parameters over all time intervals. We denote these two variants with varying number of intervals as *Tensor-#-Last* and *Tensor-#-Avg*.




(a) Convergence by iterations (b) Convergence by #pairs (c) Convergence by #events

(d) Scalability (e) Item recommendation (f) Returning-time prediction

Figure 5.3: Estimation error (a) by #iterations, (b) by #entries (1,000 events per entry), and (c) by #events per entry (10,000 entries); (d) scalability by #entries (1,000 events per entry, 500 iterations); (e) MAE of the predicted ranking; and (f) MAE of the predicted returning time.

### 5.5.2 Synthetic data

We generate two 1,024-by-1,204 user-item matrices $\mathbf{\Lambda}_0$ and $\boldsymbol{A}$ with rank five as the ground-truth. For each user-item pair, we simulate 1,000 events by algorithm D.1 with an exponential triggering kernel and get 100 million events in total. The bandwidth for the triggering kernel is fixed to one. By theorem 5.2, it is inefficient to directly estimate the exact value of the threshold value for $\rho$. Instead, we tune $\rho$, $\lambda$ and $\beta$ to give the best performance.

*How does our algorithm converge?* Figure 5.3(a) shows that it only requires a few hundred iterations to descend to a decent error for both $\mathbf{\Lambda}_0$ and $\boldsymbol{A}$, indicating algorithm 5.1 converges very fast. Since the true parameters are low-rank, Figure 5.3(b-c) verify that it only requires a modest number of observed entries, each of which induces a small number of events (1,000) to achieve a good estimation performance.

Figure 5.3(d) further illustrates that algorithm 5.1 scales linearly as the training set grows.

*What is the predictive performance?* Figure 5.3(e-f) confirm that algorithm 5.1 achieves the best predictive performance compared to other baselines. In Figure 5.3(e), all temporal methods outperform the static SVD since this classic baseline does not consider the underlying temporal dynamics of the observed sequences. In contrast, although the Poisson regression also produces static rankings of the items, it is equivalent to recommending the most popular items over time. This simple heuristic can still give competitive performance. In Figure 5.3(f), since the occurrence of a new event depends on the whole past history instead of the last one, the performance of STiC deteriorates vastly. The other tensor methods predict the returning time with the information from different time intervals. However, because our method automatically adapts different contributions of each past event to the prediction of the next event, it can achieve the best prediction performance overall.

### 5.5.3 Real Data

We also evaluate the proposed method on real datasets. *last.fm* consists of the music streaming logs between 1,000 users and 3,000 artists. There are around 20,000 observed user-artist pairs with more than one million events in total. *tmall.com* contains around 100K shopping events between 26,376 users and 2,563 stores. The unit time for both dataset is hour. MIMIC II medical dataset is a collection of de-identified clinical visit records of Intensive Care Unit patients for seven years. We filtered out 650 patients and 204 diseases. Each event records the time when a patient was diagnosed with a specific disease. The time unit is week. All model parameters $\rho$, $\lambda$, $\beta$, the kernel bandwidth and the latent rank of other baselines are tuned to give the best performance.

*Does the history help?* Because the true temporal dynamics governing the event

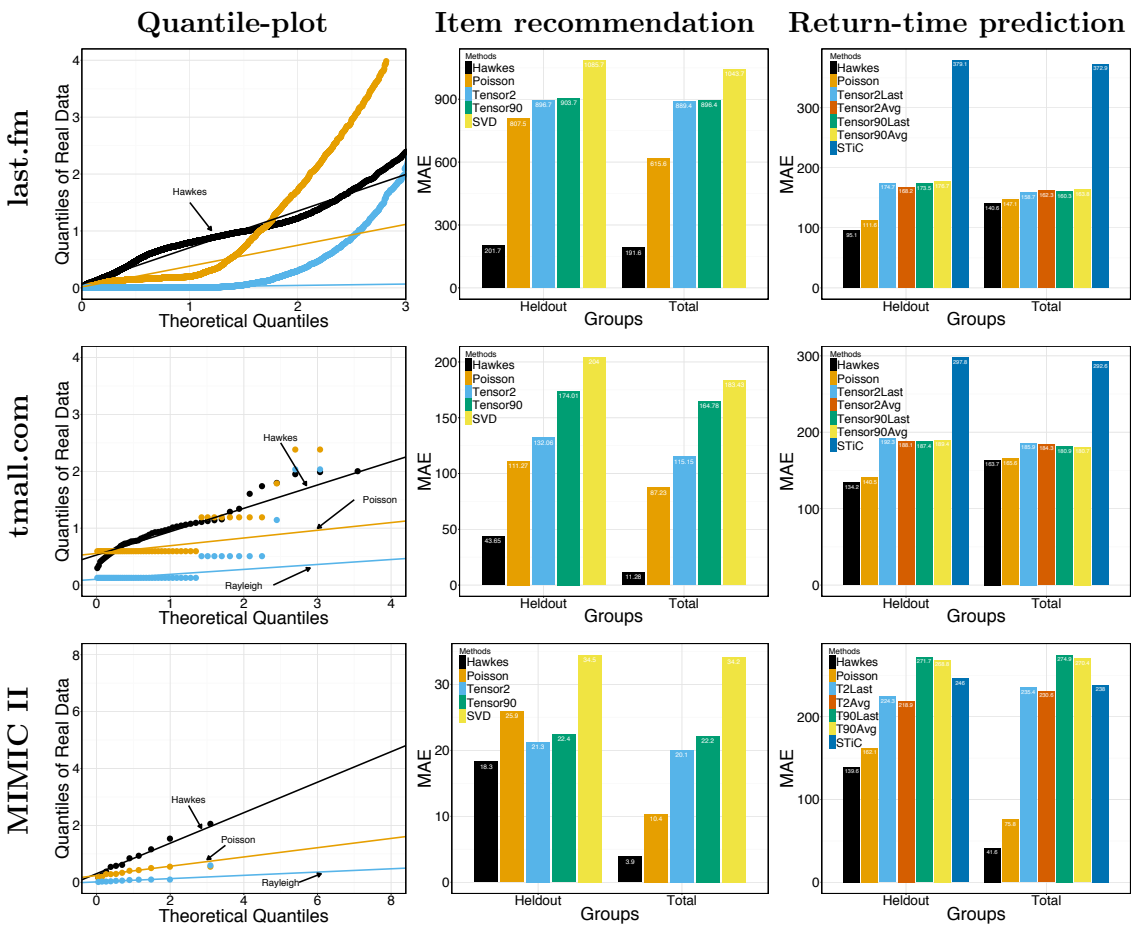



Figure 5.4: The quantile plots of different fitted processes, the MAE of predicted rankings and returning-time on the *last.fm* (top), *tmall.com* (middle) and the MIMIC II (bottom), respectively.

patterns are unobserved, we first investigate whether our model assumption is reasonable. Our *Hawkes* model considers the self-exciting effects from past user activities, while the survival analysis applied in [84] assumes *i.i.d.* inter-event gaps which might conform to an exponential (Poisson process) or Rayleigh distribution. According to the time-change theorem [35], given a sequence $\mathcal{T} = \{t_1, \ldots, t_n\}$ and a particular point process with intensity $\lambda(t)$, the set of samples $\left\{\int_{t_i-1}^{t_i} \lambda(t)dt\right\}_{i=1}^{n}$ should conform to a unit-rate exponential distribution if $\mathcal{T}$ is truly sampled from the process. Therefore, we compare the theoretical quantiles from the exponential distribution with the fittings of different models to a real sequence of (listening/shopping/visiting) events.

The closer the slope goes to one, the better a model matches the event patterns. Figure 5.4 clearly shows that our *Hawkes* model can better explain the observed data compared to the other survival analysis models.

*What is the predictive performance?* Finally, we evaluate the prediction accuracy in the $2nd$ and $3rd$ column of Figure 5.4. Since holding-out an entire testing sequence is more challenging, the performance on the *Heldout* group is a little lower than that on the average *Total* group. However, across all cases, since the proposed model is able to better capture the temporal dynamics of the observed sequences of events, it can achieve a better performance on both tasks in the end.

## 5.6 Summary

We propose the low-rank multivariate point process as one important component of our probabilistic framework for studying the recurrent events between users and items. We have developed a novel convex formulation and an efficient learning algorithm to recommend relevant services at any given moment, and to predict the next returning-time of users to existing services. Empirical evaluations on large synthetic and real data demonstrate its superior scalability and predictive performance. Moreover, our optimization algorithm can be used for solving general nonnegative matrix rank minimization problem with other convex losses under mild assumptions, which may be of independent interest.

In the next chapter, we continue to illustrate the capability of our proposed framework to model recurrent events but focus on the interactions among users and the respective inference tasks. That is, based on the proposed user activity model, how can estimate the average engagement level of users? and how can we shape the user activities to certain predefined levels?

# CHAPTER VI

# BOOSTING RECURRENT USER ACTIVITIES

In online systems, users' activities (events) can be triggered by *endogeneous* events, where users just respond to the actions of their neighbors within the network. Examples include keep responding to other people's comments to form some kind of discussions under the original post. In other cases, users' activities are simply caused by *exogenous* events, where they take actions due to drives external to the system. For instances, active users regularly share their opinions about certain conflicting topics in society to gain attentions from other online users. Active engagement of users is crucial for the healthy growth of business for modern web companies. So, how can we make inference about the engagement level based on the learned model? How much external drive should be provided to each user, such that the network activity can be steered towards a target state? In this chapter, we present the component of our framework for modeling both endogenous and exogenous event intensities, and derive a time dependent linear relation between the intensity of exogenous events and the overall network activity. Exploiting this connection, we develop a convex optimization framework for determining the required level of external drive in order for the network to reach a desired activity level. We experimented with event data gathered from Twitter, and show that our method can steer the activity of the network more accurately than alternatives.

## 6.1 Introduction

Online social platforms routinely track and record a large volume of event data, which may correspond to the usage of a service (*e.g.*, url shortening service, bit.ly). These events can be categorized roughly into *endogenous* events, where users just respond

to the actions of their neighbors within the network, or *exogenous* events, where users take actions due to drives external to the network. For instance, a user's tweets may contain links provided by bit.ly, either due to his forwarding of a link from his friends, or due to his own initiative to use the service to create a new link.

Can we model and exploit these data to steer the online community to a desired activity level? Specifically, can we drive the overall usage of a service to a certain level (*e.g.*, at least twice per day per user) by incentivizing a small number of users to take more initiatives? What if the goal is to make the usage level of a service more homogeneous across users? What about maximizing the overall service usage for a target group of users? Furthermore, these *activity shaping* problems need to be addressed by taking into account budget constraints, since incentives are usually provided in the form of monetary or credit rewards.

Activity shaping problems are significantly more challenging than traditional influence maximization problems, which aim to identify a set of users, who, when convinced to adopt a product, shall influence others in the network and trigger a large cascade of adoptions [86, 138]. First, in influence maximization, the state of each user is often assumed to be binary, either adopting a product or not [86, 28, 139, 43]. However, such assumption does not capture the recurrent nature of product usage, where the frequency of the usage matters. Second, while influence maximization methods identify a set of users to provide incentives, they do not typically provide a quantitative prescription on how much incentive should be provided to each user. Third, activity shaping concerns about a larger variety of target states, such as minimum activity requirement and homogeneity of activity, not just activity maximization.

We proceed as follows: in Section 6.2, we will address the activity shaping problems using multivariate Hawkes processes [72, 106], which can model both endogenous and exogenous recurrent social events, and were shown to be a good fit for such data in a number of recent works (*e.g.*, [23, 175, 176, 80, 105, 159]). In Section 6.3, we go

beyond model fitting, and derive a novel predictive formula for the overall network activity given the intensity of exogenous events in individual users, using a connection between the processes and branching processes [38, 135, 162, 177]. Section 6.4 presents a series of algorithms to address a diverse range of activity shaping problems given budget constraints. Compared to previous methods for influence maximization, these algorithms can provide more fine-grained control of network activity, not only steering the network to a desired steady-state activity level but also do so in a time-sensitive fashion. For example, we are able to answer complex time-sensitive queries, such as, which users should be incentivized, and by how much, to steer a set of users to use a product twice per week after one month? In Section 6.5, we provide an efficient gradient based optimization algorithm, where the matrix exponential needed for gradient computation is approximated using the truncated Taylor series expansion [7]. This algorithm allows us to validate our framework in a variety of activity shaping tasks and scale up to networks with tens of thousands of nodes. We then conduct experiments on a network of 60,000 Twitter users and more than 7,500,000 uses of a popular url shortening service showing that our algorithm can shape the network behavior much more accurately in Section 6.6, and summarize the major contributions in Section 6.7.

## 6.2 Modeling Endogenous-Exogenous Recurrent Social Events

Recurrent endogenous events often exhibit the characteristics of self-excitation, where a user tends to repeat what he has been doing recently, and mutual-excitation, where a user simply follows what his neighbors are doing due to peer pressure. These social phenomena have been made analogy to the occurrence of earthquake [111] and the spread of epidemics [170], and can be well-captured by multivariate Hawkes processes [106] as shown in a number of recent works (*e.g.*, [23, 175, 176, 80, 105, 159]).

Table 6.1: Table of symbols

| Symbol | Description |
|---|---|
| $\boldsymbol{N}(t)$ | $n$-dimensional counting process |
| $\boldsymbol{N}^{(k)}(t)$ | $n$-dimensional counting process at the $k$-generation |
| $\boldsymbol{\lambda}(t)$ | $n$-dimensional intensity function |
| $\boldsymbol{\lambda}^{(k)}(t)$ | $n$-dimensional intensity function of a counting process $\boldsymbol{N}^{(k)}(t)$ |
| $\boldsymbol{\lambda}^{(0)}(t)$ | $n$-dimensional exogenous intensity function |
| $\boldsymbol{\lambda}^{*}(t)$ | $n$-dimensional endogenous intensity function |
| $\boldsymbol{\mu}(t)$ | $n$-dimensional average intensity function at time $t$ |
| $\mathcal{H}_t$ | History of past events up to but not including time $t$ |
| $\boldsymbol{A}$ | $\boldsymbol{A} = (a_{uu'})_{u,u' \in [n]}$ excitation matrix |
| $g(t)$ | Nonnegative tiggering kernel |
| $\boldsymbol{G}(t)$ | $m \times m$ time-varying matrix $\boldsymbol{G}(t) = (a_{uu'} g(t))_{u,u' \in [m]}$ |
| $\boldsymbol{G}^{(\star k)}$ | Auto-convolution matrix |
| $U(\cdot)$ | Concave utility function |
| $C$ | Budget constraint |

Specifically, we first denote the vectorized intensity function across the $n$ dimensions as $\boldsymbol{\lambda}(t) = (\lambda_1(t), \ldots, \lambda_n(t))^\top \geqslant 0$, where each component is the respective intensity value at time $t$ for each dimension. To model the presence of both endogenous and exogenous events, we will decompose the intensity into two terms

$$\underbrace{\boldsymbol{\lambda}(t)}_{\text{overall event intensity}} = \underbrace{\boldsymbol{\lambda}^{(0)}(t)}_{\text{exogenous event intensity}} + \underbrace{\boldsymbol{\lambda}^{*}(t)}_{\text{endogenous event intensity}}, \tag{6.1}$$

where the exogenous event intensity models drive outside the network, and the endogenous event intensity models interactions within the network. We assume that hosts of social platforms can potentially drive up or down the exogenous events intensity by providing incentives to users; while endogenous events are generated due to users' own interests or under the influence of network peers, and the hosts do not interfere with them directly. We model the events generated by $n$ users in a social network as a $m$-dimensional counting process $\boldsymbol{N}(t) = (N_1(t), N_2(t), \ldots, N_n(t))^\top$, where $N_i(t)$ records the total number of events generated by user $i$ up to time $t$. Furthermore, we represent each event as a tuple $(u_i, t_i)$, where $u_i$ is the user identity and $t_i$ is the event

timing. Let the history of the process up to time $t$ be $\mathcal{H}_t := \{(u_i, t_i) \,|\, t_i < t\}$. Then, the expectation of the increment of the process, $d\boldsymbol{N}(t)$, in an infinitesimal window $[t, t + dt]$, can be expressed by

$$\mathbb{E}[d\boldsymbol{N}(t)|\mathcal{H}_t] = \boldsymbol{\lambda}(t)\, dt. \tag{6.2}$$

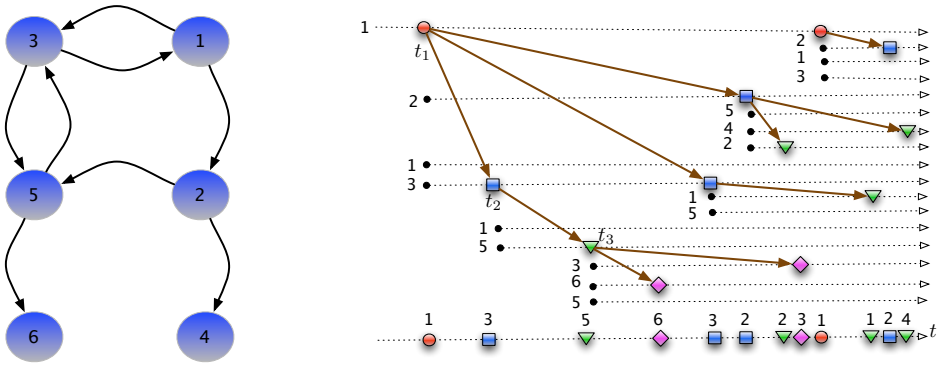
### 6.2.1 Multivariate Hawkes Process

With the multivariate Hawkes process, we assume that the exogenous event intensity is independent of the history and time, *i.e.*, $\boldsymbol{\lambda}^{(0)}(t) = \boldsymbol{\lambda}^{(0)}$ for simplicity. The strength of influence between users is parameterized by a sparse nonnegative *influence matrix* $\boldsymbol{A} = (a_{uu'})_{u,u' \in [n]}$, where $a_{uu'} > 0$ means user $u'$ directly excites user $u$. We also allow $\boldsymbol{A}$ to have nonnegative diagonals to model self-excitation of a user. Then, the intensity of the $u$-th dimension is

$$\lambda_u^*(t) = \sum\nolimits_{i:t_i<t} a_{uu_i}\, g(t - t_i) = \sum\nolimits_{u' \in [m]} a_{uu'} \int_0^t g(t - s)\, dN_{u'}(s), \tag{6.3}$$

where $g(s)$ is a nonnegative kernel function such that $g(s) = 0$ for $s \leq 0$ and $\int_0^\infty g(s)\, ds < \infty$; the second equality is obtained by grouping events according to users and use the fact that $\int_0^t g(t - s)\, dN_{u'}(s) = \sum_{u_i = u', t_i < t} g(t - t_i)$. Intuitively, $\lambda_u^*(t)$ models the propagation of peer influence over the network — each event $(u_i, t_i)$ occurred in the neighbor of a user will boost her intensity by a certain amount which itself decays over time. Thus, the more frequent the events occur in the user's neighbor, the more likely she will be persuaded to generate a new event.

Without loss of generality, we will focus on an exponential kernel, $g(t - t_i) = \exp(-\omega(t - t_i))$ in the reminder of the paper. However, multivariate Hawkes processes and the branching processed explained in next section is independent of the kernel choice and can be extended to other kernels such as power-law, Rayleigh or any other long tailed distribution over nonnegative real domain. Furthermore, we can rewrite




(a) An example social network (b) Branching structure of events

Figure 6.1: (a) an example social network where each directed edge indicates that the target node *follows*, and can be influenced by, the source node. The activity in this network is modeled using Hawkes processes, which result in branching structure of events in (b). Each exogenous event is the root node of a branch (*e.g.*, top left most red circle at $t_1$), and it occurs due to a user's own initiative; and each event can trigger one or more endogenous events (blue square at $t_2$). The new endogenous events can create the next generation of endogenous events (green triangles at $t_3$), and so forth. The social network in (a) will constrain the branching structure of events in (b), since an event produced by a user (*e.g.*, user 1) can only trigger endogenous events in the same user or one or more of her followers (*e.g.*, user 2 or user 3).

equation (6.3) in vectorial format

$$\boldsymbol{\lambda}^*(t) = \int_0^t \boldsymbol{G}(t-s)\, d\boldsymbol{N}(s), \tag{6.4}$$

by defining a $m \times m$ time-varying matrix $\boldsymbol{G}(t) = (a_{uu'} g(t))_{u,u' \in [m]}$. Note that, for multivariate Hawkes processes, the intensity, $\boldsymbol{\lambda}(t)$, itself is a random quantity, which depends on the history $\mathcal{H}_t$. We denote the expectation of the intensity with respect to history as

$$\boldsymbol{\mu}(t) := \mathbb{E}_{\mathcal{H}_t}\left[\boldsymbol{\lambda}(t)\right]. \tag{6.5}$$

### 6.2.2 Connection to Branching Processes

A branching process is a Markov process that models a population in which each individual in generation $k$ produces some random number of individuals in generation $k+1$, according some distribution [66]. In this section, we will conceptually assign

both exogenous events and endogenous events in the multivariate Hawkes process to levels (or generations), and associate these events with a branching structure which records the information on which event triggers which other events (see Figure 6.1 for an example). Note that this genealogy of events should be interpreted in probabilistic terms and may not be observed in actual data. Such connection has been discussed in Hawkes' original paper on one dimensional Hawkes processes [72], and it has recently been revisited in the context of multivariate Hawkes processes by [105].

The branching structure will play a crucial role in deriving a novel link between the intensity of the exogenous events and the overall network activity. More specifically, we assign all exogenous events to the zero-th generation, and record the number of such events as $\boldsymbol{N}^{(0)}(t)$. These exogenous events will trigger the first generation of endogenous events whose number will be recorded as $\boldsymbol{N}^{(1)}(t)$. Next these first generation of endogenous events will further trigger a second generation of endogenous events $\boldsymbol{N}^{(2)}(t)$, and so on and so forth. Then the total number of events in the network is the sum of the numbers of events from all generations

$$\boldsymbol{N}(t) = \boldsymbol{N}^{(0)}(t) + \boldsymbol{N}^{(1)}(t) + \boldsymbol{N}^{(2)}(t) + \dots \tag{6.6}$$

Furthermore, denote all events in generation $k-1$ as $\mathcal{H}_t^{(k-1)}$. Then, independently for each event $(u_i, t_i) \in \mathcal{H}_t^{(k-1)}$ in generation $k-1$, it triggers a Poisson process in its neighbor $u$ independently with intensity $a_{uu_i} g(t-t_i)$. Due to the additivity of independent Poisson processes [88], the intensity, $\lambda_u^{(k)}(t)$, of events at node $u$ and generation $k$ is simply the sum of conditional intensities of the Poisson processes triggered by all its neighbors, *i.e.*, $\lambda_u^{(k)}(t) = \sum_{(u_i,t_i)\in\mathcal{H}_t^{(k-1)}} a_{uu_i} g(t-t_i) = \sum_{u'\in[m]} \int_0^t g(t-s)\, d\boldsymbol{N}_{u'}^{(k-1)}(s)$. Concatenate the intensity for all $u \in [m]$, and use the time-varying matrix $\boldsymbol{G}(t)$ in Equation (6.4), we have

$$\boldsymbol{\lambda}^{(k)}(t) = \int_0^t \boldsymbol{G}(t-s)\, d\boldsymbol{N}^{(k-1)}(s), \tag{6.7}$$

where $\boldsymbol{\lambda}^{(k)}(t) = (\lambda_1^{(k)}(t), \dots, \lambda_m^{(k)}(t))^\top$ is the intensity for counting process $\boldsymbol{N}^{(k)}(t)$

at $k$-th generation. Again, due to the additivity of independent Poisson processes, we can decompose the intensity of $\boldsymbol{N}(t)$ into a sum of conditional intensities from different generation

$$\boldsymbol{\lambda}(t) = \boldsymbol{\lambda}^{(0)}(t) + \boldsymbol{\lambda}^{(1)}(t) + \boldsymbol{\lambda}^{(2)}(t) + \dots \tag{6.8}$$

Next, based on the above decomposition, we will develop a closed form relation between the expected intensity $\boldsymbol{\mu}(t) = \mathbb{E}_{\mathcal{H}_{t-}}[\boldsymbol{\lambda}(t)]$ and the intensity, $\boldsymbol{\lambda}^{(0)}(t)$, of the exogenous events. This relation will form the basis of our later solution for activity shaping.

## 6.3 Linking Exogenous Event Intensity to Overall Network Activity

We proceed to first link the expected intensity $\boldsymbol{\mu}^{(k)}(t) := \mathbb{E}_{\mathcal{H}_{t-}}[\boldsymbol{\lambda}^{(k)}(t)]$ of events at the $k$-th generation with $\boldsymbol{\lambda}^{(0)}(t)$, and then derive a close form for the infinite series sum

$$\boldsymbol{\mu}(t) = \boldsymbol{\mu}^{(0)}(t) + \boldsymbol{\mu}^{(1)}(t) + \boldsymbol{\mu}^{(2)}(t) + \dots \tag{6.9}$$

Define a series of auto-convolution matrices, one for each generation, with $\boldsymbol{G}^{(\star 0)}(t) = \boldsymbol{I}$ and

$$\boldsymbol{G}^{(\star k)}(t) = \int_0^t \boldsymbol{G}(t-s)\, \boldsymbol{G}^{(\star k-1)}(s)\, ds = \boldsymbol{G}(t) \star \boldsymbol{G}^{(\star k-1)}(t) \tag{6.10}$$

Then the expected intensity of events at the $k$-th generation is related to exogenous intensity $\boldsymbol{\lambda}^{(0)}$ by

**Lemma 6.1** $\boldsymbol{\mu}^{(k)}(t) = \boldsymbol{G}^{(\star k)}(t)\, \boldsymbol{\lambda}^{(0)}$.

Next, by summing together all auto-convolution matrices,

$$\boldsymbol{\Psi}(t) := \boldsymbol{I} + \boldsymbol{G}^{(\star 1)}(t) + \boldsymbol{G}^{(\star 2)}(t) + \dots$$

we obtain a linear relation between the expected intensity of the network and the intensity of the exogenous events, *i.e.*, $\boldsymbol{\mu}(t) = \boldsymbol{\Psi}(t)\boldsymbol{\lambda}^{(0)}$. The entries in the matrix

$\boldsymbol{\Psi}(t)$ roughly encode the "influence" between pairs of users. More precisely, the entry $\boldsymbol{\Psi}_{uv}(t)$ is the expected intensity of events at node $u$ due to a unit level of exogenous intensity at node $v$. We can also derive several other useful quantities from $\boldsymbol{\Psi}(t)$. For example, $\boldsymbol{\Psi}_{\bullet v}(t) := \sum_u \boldsymbol{\Psi}_{uv}(t)$ can be thought of as the overall influence user $v$ on has on all users. Surprisingly, for exponential kernel, the infinite sum of matrices results in a closed form using matrix exponentials. First, let $\widehat{\cdot}$ denote the Laplace transform of a function, and we have the following intermediate results on the Laplace transform of $\boldsymbol{G}^{(\star k)}(t)$.

**Lemma 6.2** $\widehat{\boldsymbol{G}}^{(\star k)}(z) = \int_0^\infty \boldsymbol{G}^{(\star k)}(t)\, dt = \frac{1}{z} \cdot \frac{\boldsymbol{A}^k}{(z+\omega)^k}$

With Lemma 6.2, we are in a position to prove our main theorem below:

**Theorem 6.3** $\boldsymbol{\mu}(t) = \boldsymbol{\Psi}(t)\boldsymbol{\lambda}^{(0)} = \left(e^{(\boldsymbol{A}-\omega\boldsymbol{I})t} + \omega(\boldsymbol{A}-\omega\boldsymbol{I})^{-1}(e^{(\boldsymbol{A}-\omega\boldsymbol{I})t} - \boldsymbol{I})\right)\boldsymbol{\lambda}^{(0)}$.

Theorem 6.3 provides us a linear relation between exogenous event intensity and the expected overall intensity at any point in time but not just stationary intensity. The significance of this result is that it allows us later to design a diverse range of convex programs to determine the intensity of the exogenous event in order to achieve a target intensity.

In fact, we can recover the previous results in the stationary case as a special case of our general result. More specifically, a multivariate Hawkes process is stationary if the spectral radius

$$\boldsymbol{\Gamma} := \int_0^\infty \boldsymbol{G}(t)\, dt = \left(\int_0^\infty g(t)\, dt\right)\left(a_{uu'}\right)_{u,u'\in[m]} = \frac{\boldsymbol{A}}{\omega} \tag{6.11}$$

is strictly smaller than one [106]. In this case, the expected intensity is $\boldsymbol{\mu} = (\boldsymbol{I} - \boldsymbol{\Gamma})^{-1}\boldsymbol{\lambda}^{(0)}$ independent of the time. We can obtain this relation from theorem 6.3 if we let $t \to \infty$.

**Corollary 6.4** $\boldsymbol{\mu} = (\mathbf{I} - \boldsymbol{\Gamma})^{-1}\boldsymbol{\lambda}^{(0)} = \lim_{t\to\infty} \boldsymbol{\Psi}(t)\,\boldsymbol{\lambda}^{(0)}$.

## 6.4 Convex Activity Shaping Formulation

Given the linear relation between exogenous event intensity and expected overall event intensity, we now propose a convex optimization framework for a variety of activity shaping tasks. In all tasks discussed below, we will optimize the exogenous event intensity $\boldsymbol{\lambda}^{(0)}$ such that the expected overall event intensity $\boldsymbol{\mu}(t)$ is maximized with respect to some concave utility $U(\cdot)$ in $\boldsymbol{\mu}(t)$, *i.e.*,

$$\begin{array}{ll} \text{maximize}_{\boldsymbol{\mu}(t),\boldsymbol{\lambda}^{(0)}} & U(\boldsymbol{\mu}(t)) \\ \text{subject to} & \boldsymbol{\mu}(t) = \boldsymbol{\Psi}(t)\boldsymbol{\lambda}^{(0)}, \quad \boldsymbol{c}^\top \boldsymbol{\lambda}^{(0)} \leqslant C, \quad \boldsymbol{\lambda}^{(0)} \geqslant 0 \end{array} \tag{6.12}$$

where $\boldsymbol{c} = (c_1, \ldots, c_m)^\top \geqslant 0$ is the cost per unit event for each user and $C$ is the total budget. Additional regularization can also be added to $\boldsymbol{\lambda}^{(0)}$ either to restrict the number of incentivized users (with $\ell_0$ norm $\|\boldsymbol{\lambda}^{(0)}\|_0$), or to promote a sparse solution (with $\ell_1$ norm $\|\boldsymbol{\lambda}^{(0)}\|_1$, or to obtain a smooth solution (with $\ell_2$ regularization $\|\boldsymbol{\lambda}^{(0)}\|_2$). We next discuss several instances of the general framework which achieve different goals (their constraints remain the same and hence omitted).

**Capped Activity Maximization.** In real networks, there is an upper bound (or a cap) on the activity each user can generate due to limited attention of a user. For example, a Twitter user typically posts a limited number of shortened urls or retweets a limited number of tweets [57]. Suppose we know the upper bound, $\alpha_u$, on a user's activity, *i.e.*, how much activity each user is willing to generate. Then we can perform the following *capped activity maximization* task

$$\text{maximize}_{\boldsymbol{\mu}(t),\boldsymbol{\lambda}^{(0)}} \quad \sum_{u\in[m]} \min\left\{\mu_u(t), \alpha_u\right\} \tag{6.13}$$

**Minimax Activity Shaping.** Suppose our goal is instead maintaining the activity of each user in the network above a certain minimum level, or, alternatively make the user with the minimum activity as active as possible. Then, we can perform the

following *minimax activity shaping* task

$$\text{maximize}_{\boldsymbol{\mu}(t),\boldsymbol{\lambda}^{(0)}} \quad \min_u \ \mu_u(t) \tag{6.14}$$

**Least-Squares Activity Shaping.** Sometimes we want to achieve a pre-specified target activity levels, $\boldsymbol{v}$, for users. For example, we may like to divide users into groups and desire a different level of activity in each group. Inspired by these examples, we can perform the following *least-squares activity shaping* task

$$\text{maximize}_{\boldsymbol{\mu}(t),\boldsymbol{\lambda}^{(0)}} \quad -\|\boldsymbol{B}\boldsymbol{\mu}(t) - \boldsymbol{v}\|_2^2 \tag{6.15}$$

where $\boldsymbol{B}$ encodes potentially additional constraints (*e.g.*, group partitions). Besides Euclidean distance, the family of Bregman divergences can be used to measure the difference between $\boldsymbol{B}\boldsymbol{\mu}(t)$ and $\boldsymbol{v}$ here. That is, given a function $f(\cdot) : \mathbb{R}^m \mapsto \mathbb{R}$ convex in its argument, we can use $D(\boldsymbol{B}\boldsymbol{\mu}(t)\|\boldsymbol{v}) := f(\boldsymbol{B}\boldsymbol{\mu}(t)) - f(\boldsymbol{v}) - \langle \nabla f(\boldsymbol{v}), \boldsymbol{B}\boldsymbol{\mu}(t) - \boldsymbol{v} \rangle$ as our objective function.

**Activity Homogenization.** Many other concave utility functions can be used. For example, we may want to steer users activities to a more homogeneous profile. If we measure homogeneity of activity with Shannon entropy, then we can perform the following activity homogenization task

$$\text{maximize}_{\boldsymbol{\mu}(t),\boldsymbol{\lambda}^{(0)}} \quad -\sum_{u\in[m]} \mu_u(t) \ln \mu_u(t). \tag{6.16}$$

## 6.5 Efficient Implementation

The activity shaping problems defined above requires efficient evaluation of matrix exponentials for computing $\boldsymbol{\Psi}(t)$, the instantaneous average intensity at time $t$. For medium scale dense matrix, we can leverage well-known numerical methods to compute matrix exponentials [55]. However, in large networks with sparse graph structure $\boldsymbol{A}$, the explicit computation of $\boldsymbol{\Psi}(t)$ quickly become intractable. Fortunately, a key

property of our convex activity shaping framework is that the gradient computation in each formulation only depends on $\boldsymbol{\Psi}(t)$ through matrix-vector product operations. To elaborate, use Theorem 6.3 we rewrite the multiplication of $\boldsymbol{\Psi}(t)$ and a vector $\boldsymbol{v}$ as $\boldsymbol{\Psi}(t)\boldsymbol{v} = e^{(\boldsymbol{A}-\omega\boldsymbol{I})t}\boldsymbol{v} + \omega(\boldsymbol{A}-\omega\boldsymbol{I})^{-1}\left(e^{(\boldsymbol{A}-\omega\boldsymbol{I})t}\boldsymbol{v} - \boldsymbol{v}\right)$ Therefore, we first compute $e^{(\boldsymbol{A}-\omega\boldsymbol{I})t}\boldsymbol{v}$, subtract $\boldsymbol{v}$ from it; and then solve a sparse linear system of equations, $\omega(\boldsymbol{A}-\omega\boldsymbol{I})x = \left(e^{(\boldsymbol{A}-\omega\boldsymbol{I})t}\boldsymbol{v} - \boldsymbol{v}\right)$, to find $x$. The final vector is then result of adding $x$ and $e^{(\boldsymbol{A}-\omega\boldsymbol{I})t}\boldsymbol{v}$ (which is computed at the first step). Therefore, an efficient computation will involve two ingredients: solving sparse linear system of equation and multiplying a matrix exponential with a vector. For both parts, we elaborate on two very efficient algorithms for solving a sparse linear system of equations and for computing the product of matrix exponential with a vector.

We use the well-known GMRES method for solving the linear system [141], which is an Arnoldi process for constructing an $l_2$-orthogonal basis of Krylov subspaces. It solves the linear system by iteratively minimizing the norm of the residual vector over a Krylov subspace. For the second part we turn to the iterative algorithm of Al-Mohy et al. [7] which combines a scaling and squaring method with a truncated Taylor series approximation to the matrix exponential.

With efficient computation of the gradients, we then apply the the projected gradient descent [24] optimization framework with the following gradient for each activity shaping formation. *(i)* Activity maximization[1]: $\boldsymbol{g}(\boldsymbol{\lambda}^{(0)}) = \boldsymbol{\Psi}(t)^\top\boldsymbol{v}$, where $\boldsymbol{v}$ is defined such that $v_j = 1$ if $\alpha_j > \mu_j$, and $v_j = 0$, otherwise. *(ii)* Minimax Activity Shaping: $\boldsymbol{g}(\boldsymbol{\lambda}^{(0)}) = \boldsymbol{\Psi}(t)^\top\boldsymbol{e}$, where $\boldsymbol{e}$ is defined such that $e_j = 1$ if $\mu_j = \mu_{min}$, and $e_j = 0$, otherwise. *(iii)* Least-square activity shaping: $\boldsymbol{g}(\boldsymbol{\lambda}^{(0)}) = 2\boldsymbol{\Psi}(t)^\top\boldsymbol{B}^\top\left(\boldsymbol{B}\boldsymbol{\Psi}(t)\boldsymbol{\lambda}^{(0)} - \boldsymbol{v}\right)$. *(iv)* Activity homogenization: $\boldsymbol{g}(\boldsymbol{\lambda}^{(0)}) = \boldsymbol{\Psi}(t)^\top \ln\left(\boldsymbol{\Psi}(t)\boldsymbol{\lambda}^{(0)}\right) + \boldsymbol{\Psi}(t)^\top\mathbf{1}$, where $\ln(\cdot)$ on a vector is the element-wise natural logarithm. Algorithm 6.1 summarizes the key steps of the procedure.

[1]For non-differential objectives, sub-gradient algorithms can be used instead.

**Algorithm 6.1:** Projected Gradient Descent for Activity Shaping

Initialize $\boldsymbol{\lambda}^{(0)}$;
**repeat**
  Project $\boldsymbol{\lambda}^{(0)}$ into the linear space $\boldsymbol{\lambda}^{(0)} \geqslant 0$, $\boldsymbol{c}^{\top}\boldsymbol{\lambda}^{(0)} \leqslant \boldsymbol{C}$;
  Evaluate the gradient $\boldsymbol{g}(\boldsymbol{\lambda}^{(0)})$ at $\boldsymbol{\lambda}^{(0)}$ using GMRES and Al-Mohy methods;
  Update $\boldsymbol{\lambda}^{(0)}$ using the gradient $\boldsymbol{g}(\boldsymbol{\lambda}^{(0)})$;
**until** *convergence*;

Table 6.2: Number of adopters and usages for each URL shortening service.

| Service | # Adopters | # Usages |
|---|---|---|
| Bitly | 55,883 | 5,046,710 |
| TinyURL | 46,577 | 1,682,459 |
| Isgd | 28,050 | 596,895 |
| TwURL | 15,215 | 197,568 |
| SnURL | 4,462 | 41,823 |
| Doiop | 88 | 643 |

## 6.6 Experiments

We evaluate our activity shaping framework using both simulated and real world held-out data, and show that our approach significantly outperforms several baselines.

### 6.6.1 Setup

We use data gathered from Twitter as reported in [25], which comprises of all public tweets posted by 60,000 users during a 8-month period, from January 2009 to September 2009. For every user, we record the times she uses any of the following six url shortening services: Bitly , TinyURL, Isgd, TwURL, SnURL, Doiop. Table 6.2 shows their details. It includes a total of 7,566,098 events (adoptions) during the 8-month period. We evaluate the performance of our framework on a subset of 2,241 active users, linked by 4,901 edges, which we call 2K dataset, and we evaluate its scalability on the overall 60,000 users, linked by $\sim$ 200,000 edges, which we call 60K dataset. The 2K dataset accounts for 691,020 url shortened service uses while the 60K dataset accounts for $\sim$7.5 million uses. Finally, we treat each service as independent

cascades of events. In the experiments, we estimated the nonnegative influence matrix $\boldsymbol{A}$ and the exogenous intensity $\boldsymbol{\lambda}^{(0)}$ using maximum log-likelihood, as in previous work [175, 176, 159]. We used a temporal resolution of one minute and selected the bandwidth $\omega = 0.1$ by cross validation. Loosely speaking, $\omega = 0.1$ corresponds to loosing 70% of the initial influence after 10 minutes, which may be explained by the rapid rate at which each user' news feed gets updated.

### 6.6.2 Evaluation

We focus on three tasks: capped activity maximization, minimax activity shaping, and least square activity shaping. We set the total budget to $C = 0.5$, which corresponds to supporting a total extra activity equal to 0.5 actions per unit time, and assume all users entail the same cost. In the capped activity maximization, we set the upper limit of each user's intensity, $\boldsymbol{\alpha}$, by adding a nonnegative random vector to their inferred initial intensity. In the least-squares activity shaping, we set $\boldsymbol{B} = \mathbf{I}$ and aim to create three groups of users, namely less-active, moderate, and super-active users. We use three different evaluation schemes, with an increasing resemblance to a real world scenario:

**Theoretical objective:** we compute the expected overall (theoretical) intensity by applying Theorem 6.3 on the optimal exogenous event intensities, $\boldsymbol{\lambda}^{(0)}_{opt}$, to each of the three activity shaping tasks, as well as the learned $\boldsymbol{A}$ and $\omega$. We then compute and report the value of the objective functions. In this evaluation scheme, we do not use the learned $\boldsymbol{\lambda}^{(0)}$.

**Simulated objective:** we simulate 50 cascades with Ogata's thinning algorithm [124], using the optimal exogenous event intensities, $\boldsymbol{\lambda}^{(0)}_{opt}$, to each of the three activity shaping tasks, and the learned $\boldsymbol{A}$ and $\omega$. We then estimate empirically the overall event

intensity based on the simulated cascades, by computing a running average over non-overlapping time windows, and report the value of the objective functions based on this estimated overall intensity. This evaluation scheme does not use the learned $\boldsymbol{\lambda}^{(0)}$ either.

**Held-out data:** The most interesting evaluation scheme would entail carrying out real interventions in a social platform. However, since this is very challenging to do, instead, in this evaluation scheme, we use held-out data to simulate such process, proceeding as follows. We first partition the 8-month data into 50 five-day long contiguous intervals. Then, we use one interval for training and the remaining 49 intervals for testing. Suppose interval 1 is used for training, the procedure is as follows:

1. We estimate $\boldsymbol{A}_1$, $\omega_1$ and $\boldsymbol{\lambda}_1^{(0)}$ using the events from interval 1. Then, we fix $\boldsymbol{A}_1$ and $\omega_1$, and estimate $\boldsymbol{\lambda}_i^{(0)}$ for all other intervals, $i = 2, \ldots, 49$.

2. Given $\boldsymbol{A}_1$ and $\omega_1$, we find the optimal exogenous event intensities, $\boldsymbol{\lambda}_{opt}^{(0)}$, for each of the three activity shaping task, by solving the associated convex program. We then sort the estimated $\boldsymbol{\lambda}_i^{(0)}$ $(i = 2, \ldots, 49)$ according to their similarity to $\boldsymbol{\lambda}_{opt}^{(0)}$, using the Euclidean distance $\|\boldsymbol{\lambda}_{opt}^{(0)} - \boldsymbol{\lambda}_i^{(0)}\|_2$.

3. We estimate the overall event intensity for each of the 49 intervals $(i = 2, \ldots, 49)$, as in the "simulated objective" evaluation scheme, and sort these intervals according to the value of their corresponding objective function.

4. Last, we compute and report the rank correlation score between the two orderings obtained in step 2 and 3.[2] The larger the rank correlation, the better the method.

[2]rank correlation = number of pairs with consistent ordering / total number of pairs.

We repeat this procedure 50 times, choosing each different interval for training once, and compute and report the average rank correlations.

### 6.6.3 Baselines

**Capped activity maximization problem.** We design the following heuristics:

- XMU allocates the budget based on users' current activity. In particular, it assigns the budget to each of the half top-most active users proportional to their average activity, $\boldsymbol{\mu}(t)$, computed from the inferred parameters.

- WEI assigns positive budget to the users proportionally to their sum of out-going influence ($\sum_u a_{uu'}$). This heuristic allows us (by comparing its results to CAM) to understand the effect of considering the whole network with respect to only consider the direct (out-going) influence.

- DEG assumes that more central users, *i.e.*, more connected users, can leverage the total activity, therefore, assigns the budget to the more connected users proportional to their degree in the network.

- PRK sorts the users according to the their pagerank in the weighted influence network ($\boldsymbol{A}$) with the damping factor set to 0.85%, and assigns the budget to the top users proportional their pagerank value.

**Minimax activity shaping.** We compare with the following baselines:

- UNI allocates the total budget equally to all users.

- MINMU divides uniformly the total budget among half of the users with lower average activity $\boldsymbol{\mu}(t)$, which is computed from the inferred parameters.

- LP finds the top half of least-active users in the current network and allocates the budget such that after the assignment the network has the highest minimum

activity possible. This method uses linear programming to learn exogenous activity of the users, but, in contrast to the proposed method, does not consider the network and propagation of adoptions.

- GRD finds the user with minimum activity, assigns a portion of the budget, and computes the resulting $\boldsymbol{\mu}(t)$. It then repeats the process to incentivize half of users.

**Least-square activity shaping.** We implement the following baselines:

- PROP shapes the activity by allocating the budget proportional to the desired shape, *i.e.*, the shape of the assignment is similar to the target shape.

- LSGRD greedily finds the user with the highest distance between her current and target activity, assigns her a budget to reach her target, and proceeds this way to consume the whole budget.

Each baseline relies on a specific property to allocate the budget (*e.g.* connectedness in DEG). However, most of them face two problems: The first one is how many users to incentivize and the second one is how much should be paid to the selected users. They usually rely on heuristics to reveal these two problems (*e.g.* allocating an amount proportional to that property and/or to the top half users sorted based on the specific property). In contrast, our framework is comprehensive enough to address those difficulties based on well-developed theoretical basis. This key factor accompanied with the appropriate properties of Hawkes process for modeling social influence (*e.g.* mutually exciting) make the proposed method the best.

### 6.6.4 Results

**Theoretical prediction.** For the experiments on simulated objective function and held-out data we have estimated intensity from the events data. In this section, we

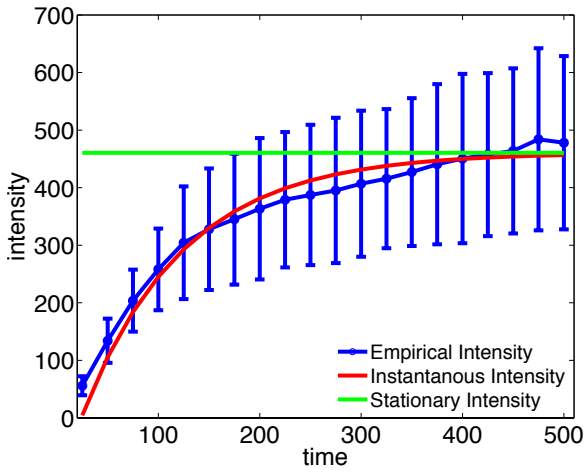



Figure 6.2: Evolution in time of empirical and theoretical intensity.

will see how this empirical intensity resembles the theoretical intensity. We generate a synthetic network over 100 users. For each user in the generated network, we uniformly sample from $[0, 0.1]$ the exogenous intensity, and the endogenous parameters $a_{uu'}$ are uniformly sampled from $[0, 0.1]$. A bandwidth $\omega = 1$ is used in the exponential kernel. Then, the intensity is estimated empirically by dividing the number of events by the length of the respective interval.

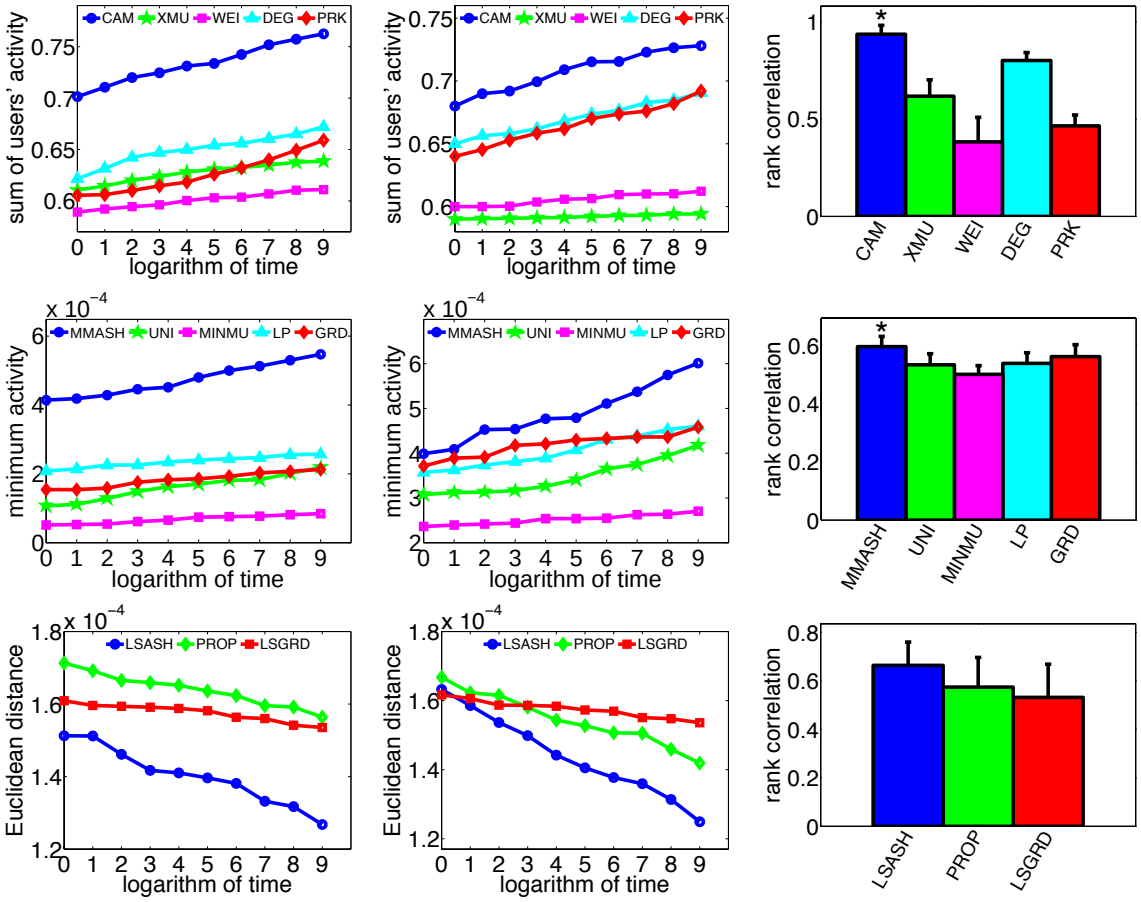
We compute the mean and variance of the empirical activity for 100 independent runs. As illustrated in Figure 6.2, the average empirical intensity (the blue curve) clearly follows the theoretical instantaneous intensity (the red curve) but, as expected, as we are further from the starting point (*i.e.*, as time increases), the standard deviation of the estimates (shown in the whiskers) increases. Additionally, the green line shows the average stationary intensity. As it is expected, the instantaneous intensity tends to the stationary value when the network has been run for sufficient long time.

**Capped activity maximization (CAM).** The first row of Figure 6.3 summarizes the results for the three different evaluation schemes. We find that our method (CAM) consistently outperforms the alternatives. For the theoretical objective, CAM is 11-% better than the second best, DEG. The difference in overall users' intensity from

DEG is about 0.8 which, roughly speaking, leads to at least an increase of about $0.8 \times 60 \times 24 \times 30 = 34,560$ in the overall number of events in a month. In terms of simulated objective and held-out data, the results are similar and provide empirical evidence that, compared to other heuristics, degree is an appropriate surrogate for influence, while, based on the poor performance of XMU, it seems that high activity does not necessarily entail being influential. To elaborate on the interpretability of the real-world experiment on held-out data, consider for example the difference in rank correlation between CAM and DEG, which is almost 0.1. Then, roughly speaking, this means that incentivizing users based on our approach accommodates with the ordering of real activity patterns in $0.1 \times \frac{50 \times 49}{2} = 122.5$ more pairs of realizations.

**Minimax activity shaping (MMASH).** The second row of Figure 6.3 summarizes the results for the three different evaluation schemes. We find that our method (MMASH) consistently outperforms the alternatives. For the theoretical objective, it is about $2\times$ better than the second best, LP. Importantly, the difference between MMASH and LP is not trifling and the least active user carries out $2 \times 10^{-4} \times 60 \times 24 \times 30 = 4.3$ more actions in average over a month. As one may have expected, GRD and LP are the best among the heuristics. The poor performance of MINMU, which is directly related to the objective of MMASH, may be because it assigns the budget to a low active user, regardless of their influence. However, our method, by cleverly distributing the budget to the users whom actions trigger many other users' actions (like those ones with low activity), it benefits from the budget most. In terms of simulated objective and held-out data, the algorithms' performance become more similar.

**Least-squares activity shaping (LSASH).** The third row of Figure 6.3 summarizes the results for the three different evaluation schemes. We find that our method




(a) Theoretical objective (b) Simulated objective (c) Held-out data

Figure 6.3: Row 1: Capped activity maximization. Row 2: Minimax activity shaping. Row 3: Least-squares activity shaping. * means statistical significant at level of 0.01 with paired t-test between our method and the second best

(LSASH) consistently outperforms the alternatives. Perhaps surprisingly, PROP, despite its simplicity, seems to perform slightly better than LSGRD. This is may be due to the way it allocates the budget to users, *e.g.*, it does not aim to strictly fulfill users' target activity but benefit more users by assigning budget proportionally.

**Scalability.** The most computationally demanding part of the proposed algorithm is the evaluation of matrix exponentials, which we scale up by utilizing techniques from matrix algebra, such as GMRES and Al-Mohy methods. As a result, we are able to run our methods in a reasonable amount of time on the 60K dataset, specifically, in

comparison with a naive implementation of matrix exponential evaluations. Specifically, The naive implementation of the algorithm requires computing the matrix exponential once, and using it in (non-sparse huge) matrix-vector multiplications,

$$T_{naive} = T_{\mathbf{\Psi}} + kT_{prod}. \tag{6.17}$$

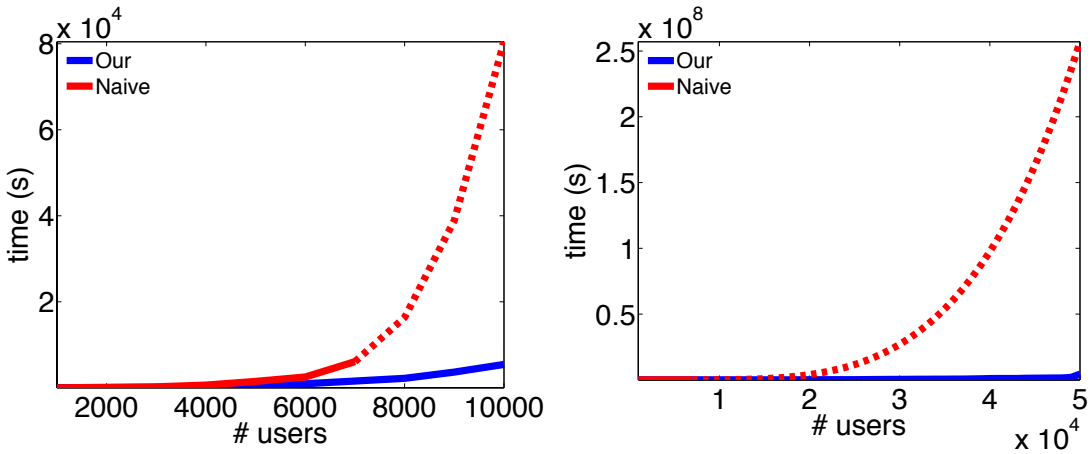
Here, $T_{\mathbf{\Psi}}$ is the time to compute $\mathbf{\Psi}(t)$, which itself comprised of three parts; matrix exponential computation, matrix inversion and matrix multiplications. $T_{prod}$ is the time for multiplication between the large non-sparse matrix and a vector plus the time to compute the inversion via solving linear systems of equation. Finally, $k$ is the number of gradient computations, or more generally, the number of iterations in any gradient-based iterative optimization. The dominant factor in the naive approach is the matrix exponential.

In contrast, the proposed framework benefits from the fact that the gradient depends on $\mathbf{\Psi}(t)$ only through matrix-vector products. Thus, the running time of our activity shaping framework will be written as

$$T_{our} = kT_{grad}, \tag{6.18}$$

where $T_{grad}$ is the time to compute the gradient which itself comprises the time required to solve a couple of linear systems of equations and the time to compute a couple of exponential matrix-vector multiplication.

Figure 6.4 demonstrates $T_{our}$ and $T_{naive}$ with respect to the number of users. For better visualization we have provided two graphs for up to 10,000 and 50,000 users, respectively. We set $k$ equal to the number of users. Since the dominant factor in the naive computation method is matrix exponential, the choice of $k$ is not that determinant. The time for computing matrix exponential is interpolated for more than 7000 users; and the interpolated total time, $T_{naive}$ , is shown in red dashed line. These experiments are done in a machine equipped with one 2.5 GHz AMD




(a) 1000 users (b) 5000 users

Figure 6.4: Scalability of least-squares activity shaping.

Opteron Processor. This graph clearly shows the significance of designing an scalable algorithm.

## 6.7 Summary

In this chapter, based on the modeling of exogenous and endogenous type of events with multivariate Hawkes process, we derive an analytic solution for making inference about the average instantaneous activity level of users over a given network at any time $t$. Then, we specify a series of activity shaping formulations with efficient algorithms seeking to incentivize users in different ways in order to effectively improve their engagement. By running experiments on real datasets, we have shown that our model can shape activities to certain desired level better than many sophisticated heuristics. To capture the recurrent events induced from user activities, we have thus far assumed a linear dependency over the history. In particular, the Hawkes process assumes that the influences from past events to the occurrence of a newly triggered event are *linearly* additive. Since the true evolutionary mechanisms are never known, it will be much more flexible and powerful to learn a general *nonlinear* dependency and incorporate the influences from other contextual information, which will be the topics in the following chapters of advanced models.

# PART III ADVANCED PROCESSES

Occasionally, in addition to time, extra information might be associated with each temporal event. Examples include the magnitude of an earthquake, the passenger pick-up place of a taxi, the buying or selling action conducted on a stock, *etc.* Such type of data usually exist in the form of covariates, and thus is often known as the markers. Furthermore, for most user generated data, such as blogs and news articles, the textual contents of these information are also accessible. As a consequence, in the third part of the thesis, we demonstrate its extensibility to incorporate other type of information in addition to time.

**Accurate Modeling:** we propose the recurrent marked temporal point process to jointly model the timing and the marker of an event by learning a general nonlinear dependency over the history based on recurrent neural networks. Moreover, we also design the Dirichlet point process by building a previously unexplored connection between Bayesian Nonparametrics and temporal point processes, which allows the number of dimensions to grow in order to accommodate the increasing complexity of online streaming data.

**Efficient Learning:** we develop an efficient stochastic gradient algorithm for learning the recurrent temporal point process. We also propose an online Bayesian updating formulation for updating the model parameters of the Dirichlet point process from document streams.

**Scalable Inference:** we propose an efficient online inference algorithm based on particle filters which can scale up to millions of news articles with near constant processing time per document and moderate memory consumption.

# CHAPTER VII

# MODELING RECURRENT MARKED EVENTS

Marked temporal point processes and intensity functions are the mathematical framework for modeling event data with covariates. However, typical point process models, such as Hawkes processes [72], continuous-time Markov chains [81], autoregressive conditional duration processes [49], often make strong assumptions about the generative processes of the event data, which may or may not reflect the reality, and the assumptions of specifically fixed parametric forms have also restricted the expressive power of the respective processes. Can we obtain a more expressive model of marked temporal point processes? How can we learn such a model from massive data?

We propose the Recurrent Marked Temporal Point Process (RMTPP), which is an important extension of our framework to jointly model the event timings and markers in this chapter. The key idea of our approach is to view the intensity function of a temporal point process as a general nonlinear function of the history, and parameterize the function with a recurrent neural network. We develop an efficient learning framework which can readily scale up to millions of events. Using both synthetic and real world datasets, we show that, in the case where the true models are parametric models, RMTPP can learn the dynamics of such models without the need to know the actual parametric forms; and in the case where the true models are unknown, RMTPP can also learn the dynamics and achieve better predictive performance than other parametric alternatives based on various prior assumptions.

## 7.1 Introduction

Event data with marker information can be produced from social activities, to financial transactions, to electronic health records, which may provide rich information

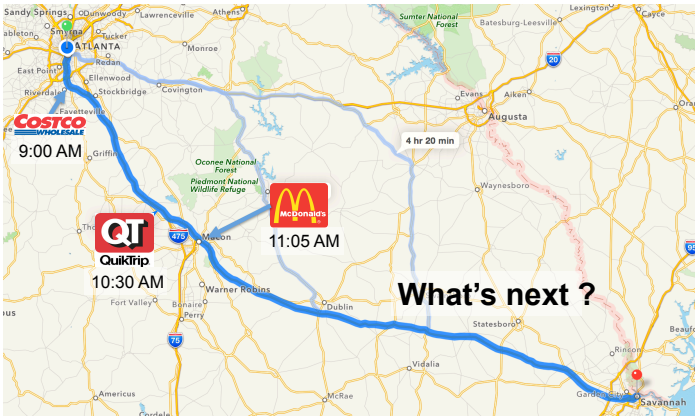



Figure 7.1: A user has visited Costco at 9:00AM, refilled the gas in QT at 10:30AM, and then had lunch in MacDonald at 11:05AM. Given the trace of these past locations and time, can we predict what the next stop will be and when it will happen in the future?

about what *type* of event is happening between *which entities* by *when* and *where*. For instance, people might visit various places at different moments of a day. Professional trading systems buy and sell large amounts of stocks within short-time frames. Patients regularly go to the clinic with a longitudinal data of diagnoses about their concerned diseases.

Although the aforementioned situations may come from a broad range of domains, we are interested in a commonly encountered question: based on the observed sequence of events, *can we predict what kind of event will take place at what time in the future*? Accurately predicting the type and the timing of the next event will have many interesting applications. For mainstream personal assistants, shown in Figure 7.1, since people tend to visit different places depending on the temporal/spatial contexts, successfully predicting their next destinations at the most likely time will make such services more relevant and usable. In stock market, accurately forecasting when to sell or buy a particular stock means critical business success. For modern health-care, patients may have several diseases that have complicated dependencies on each other. Accurately estimating when a clinical event might occur can effectively facilitate patient-specific care and prevention to reduce the potential future risks.

Existing studies in literature attempt to approach this problem mainly in two ways: first, classic varying-order Markov models [18] formulate the problem as a discrete-time sequence prediction task. Based on the observed sequence of states, they can predict the most likely state the process will evolve into on the next step. As a result, one limit of the family of classic Markov models is that it assumes the process proceeds by unit time-steps, so it cannot capture the heterogeneity of the time to predict the timing of the next event in the future. Furthermore, when the number of states is large, Markov model usually cannot capture long dependency on the history since the overall state-space will grow exponentially. Semi-Markov model [81] can model the continuous time-interval between two successive states to some extent by assuming the intervals have very simple distributions. It has the same state-space explosion issue when the order grows.

Second, marked temporal point processes and intensity functions are a more general mathematical framework for modeling such event data. For example, in seismology, marked temporal point processes have originally been widely used for modeling earthquakes and aftershocks [72, 71, 73, 125]. Each earthquake can be represented as a point in the temporal-spatial space, and seismologists have proposed different formulations to capture the randomness of these events. In the financial area, temporal point processes are active research topics of econometrics, which often leads to many simple interpretations of the complex dynamics of modern electronic markets [10, 11, 12]. However, typical point process models, such as Hawkes processes [72], continuous Markov chains [81], autoregressive conditional duration processes [49, 161, 50], are making specific assumptions about the functional forms of the generative processes, which may or may not reflect the reality, and thus the respective fixed simple parametric representations may restrict the expressive power of these models.

Therefore, we propose a novel marked temporal point process which is able to capture a flexible nonlinear dependency structure inherent in general time-series data

Table 7.1: Table of symbols

| Symbol | Description |
|---|---|
| $t_i$ | Occurrence time of the $i$-th event |
| $d_i$ | Inter-event duration $d_i = t_i - t_{i-1}$ |
| $y_i$ | Marker of the $i$-th event |
| $\mathcal{S}^i$ | The $i$-th sequence of events |
| $\mathcal{H}_{t_n}$ | History comprising the past events $\{(t_1, y_1), \ldots, (t_{n-1}, y_{n-1})\}$ |
| $f(t_i, y_i \mid \mathcal{H}_{t_i})$ | Joint density of time and marker conditioned on the history |
| $N(t)$ | Number of events up to time $t$ |
| $\boldsymbol{h}_j$ | Hidden state representing the past influence up to the $j$-th event |
| $\boldsymbol{y}_j^b$ | One-hot representation for marker $y_j$ |
| $\boldsymbol{y}_j$ | Embedded representation for marker $y_j$ |
| $\boldsymbol{t}_j$ | Temporal features extracted from $t_j$ |
| $\boldsymbol{W}_{em}$ | Weight matrix for the marker in the input layer |
| $\boldsymbol{b}_{em}$ | Bias term for the marker in the input layer |
| $\boldsymbol{W}^y$, $\boldsymbol{W}^t$, $\boldsymbol{W}^h$ | Weight matrix for the marker, the time and the previous hidden state in the hidden layer |
| $\boldsymbol{b}_h$ | Bias term in the hidden layer |
| $\boldsymbol{V}^y$, $\boldsymbol{v}^t$ | Weight matrix and vector for the marker and the time in the output layer |
| $\boldsymbol{b}^y, b^t$ | Bias terms for the marker and the time in the output layer |

without requiring any prior knowledge and assumption about the specific form of the hidden temporal dynamics. The rest of this chapter is organized as follows: in Section 7.2 we briefly review some commonly used temporal point processes and point out their respective major limits. In Section 7.3, we present our proposed Recurrent Marked Temporal Point Process to jointly model the timing and marker information. We report extensive experimental evaluations in Section 7.4, and conclude the chapter in Section 7.5.

## 7.2 Marked Temporal Point Process

Marked temporal point process is a powerful mathematical tool to model the latent mechanisms governing the observed random point patterns along time. Since the

occurrence of an event may be triggered by what happened in the past, we can essentially specify models for the timing of the next event given what we have already known so far. More formally, a marked temporal point process is a random process of which the realization consists of a list of discrete events localized in time, $\{t_j, y_j\}$, with the timing $t_j \in \mathbb{R}^+$, the marker $y_j \in \mathcal{Y}$ and $j \in \mathbb{Z}^+$ . Let the history $\mathcal{H}_t$ be the list of event time and marker pairs $\{(t_1, y_1), \ldots, (t_n, y_n)\}$ up to the time $t$. The length $d_{j+1} = t_{j+1} - t_j$ of the time interval between neighboring successive events $t_j$ and $t_{j+1}$ is referred to as the inter-event duration.

Given the history of past events, we can explicitly specify the conditional density function that the next event will happen at time $t$ with type $y$ as $f^*(t, y) = f(t, y|\mathcal{H}_t)$ where $f^*(t, y)$ emphasizes that this density is conditional on the history. By applying the chaining rule, we can derive the joint likelihood of observing a sequence as the following:

$$f\left(\{(t_j, y_j)\}_{j=1}^n\right) = \prod_j f(t_j, y_j|\mathcal{H}_t) = \prod_j f^*(t_j, y_j) \tag{7.1}$$

One can design many forms for $f^*(t_j, y_j)$. However, in practice, people typically choose very simple factorized formulations like $f(t_j, y_j|\mathcal{H}_t) = f(y_j)f(t_j|\ldots, t_{j-2}, t_{j-1})$. One can think of $f(y_j)$ as a multinomial distribution when $y_j$ can only take finite number of values. $f^*(t_j) := f(t_j|\ldots, t_{j-2}, t_{j-1})$ denotes the conditional density of the event occurring at the time $t_j$ given the sequence of past events.

### 7.2.1 Parametrizations

The temporal information in a marked point process can be well captured by a typical temporal point process as usual. We can see from previous chapters that particular functional forms of the conditional intensity function $\lambda^*(t)$ are often designed to capture the phenomena of interests. In the following, we review a few representative examples of temporal point processes where the conditional intensity has specific parametric forms.

**Poisson process [88].** The homogeneous Poisson process is the simplest point process. The inter-event times are independent and identically distributed random variables conforming to the exponential distribution. The conditional intensity function is assumed to be independent of the history $\mathcal{H}_t$ and keeps constant over time, *i.e.*, $\lambda^*(t) = \lambda_0 \geqslant 0$. For a more general inhomogeneous Poisson process, the intensity is also assumed to be independent of the history $\mathcal{H}_t$, but it can be a function varying over time, *i.e.*, $\lambda^*(t) = g(t) \geqslant 0$.

**Hawkes process [72].** A Hawkes process captures the mutual excitation phenomena among events with the conditional intensity being defined as

$$\lambda^*(t) = \gamma_0 + \alpha \sum_{t_j < t} \gamma(t, t_j), \tag{7.2}$$

where $\gamma(t, t_j) \geqslant 0$ is the triggering kernel modeling the contribution from the past event at $t_j$ to the occurrence of a new event at time $t$, $\gamma_0 \geqslant 0$ is a baseline intensity independent of the history, and the summation of kernel terms is history dependent and a stochastic process by itself. The kernel function can be chosen in advance, *e.g.*, $\gamma(t, t_j) = \exp(-|t - t_j|)$ or $\gamma(t, t_j) = \mathbb{I}[t > t_j]$, or directly learned from data.

A distinctive feature of the Hawkes process is that the occurrence of each historical event increases the intensity by a certain amount. Since the intensity function depends on the history up to time $t$, the Hawkes process is essentially a conditional Poisson process (or doubly stochastic Poisson process [87]) in the sense that conditioned on the history $\mathcal{H}_t$, the Hawkes process is a Poisson process formed by the superposition of a background homogeneous Poisson process with the intensity $\gamma_0$ and a set of inhomogeneous Poisson processes with the intensity $\gamma(t, t_j)$. However, because the events in a past interval can affect the occurrence of the events in later intervals, the Hawkes process in general is more expressive than a Poisson process.

**Self-correcting process [79].** In contrast to the Hawkes process, the self-correcting process seeks to produce regular point patterns with the conditional intensity function defined by

$$\lambda^*(t) = \exp\left(\mu t - \sum_{t_i<t} \alpha\right), \tag{7.3}$$

where $\mu > 0, \alpha > 0$. The intuition is that while the intensity increases steadily, every time when a new event appears, it is decreased by multiplying a constant $e^{-\alpha} < 1$, so the chance of new points decreases after an event has occurred recently.

**Autoregressive Conditional Duration process [49].** An alternative way of conditional intensity parametrization is to capture the dependency between inter-event timings $d_i = t_i - t_{i-1}$. The expectation for $d_i$ is given by $\psi_i = E(d_i | \dots, d_{i-2}, d_{i-1})$. The simplest form assumes that $d_i = \psi_i \epsilon_i$ where $\epsilon_i$ is independently and identically distributed exponential variables with expectation one. As a consequence, the conditional intensity has the following form:

$$\lambda^*(t) = \psi_{N(t)}^{-1}, \tag{7.4}$$

where $\psi_i = \gamma_0 + \sum_{j=0}^{m} \alpha_j d_{i-j}$ to capture the influences from the most recent $m$ durations, and $N(t)$ is the total number of events up to $t$.

### 7.2.2 Major Limitations

**Curse of Model Misspecification.** All these different parameterizations of the conditional intensity function seek to capture certain forms of dependency on the history in different ways: Poisson process makes the assumption that the duration is stationary; Hawkes process assumes that the influences from past events are linearly additive towards the current event; Self-correcting process specifies a non-linear dependency over these past events; and autoregressive conditional duration model imposes a linear structure between successive inter-event durations. These different

parameterizations *encode our prior knowledge* about the latent dynamics we try to model. In practice, however, the true model is never known. Thus, we have to try different specifications for $\lambda^*(t)$ to tune the predictive performance and most often we can expect to suffer from certain errors caused by the model misspecification.

**Marker Generation.** Furthermore, it is quite often that we have additional information (or covariates) associated with each event like the markers. For instance, the marker of a NYC taxi can be the neighborhood-name of the place when it picks up (or drops off) passengers; the marker of each financial transaction can be the action of buying (or selling); and the marker of a clinical event can be the diagnosis of the major disease. Classic temporal point processes can be extended to capture the marker information mainly in the following two ways: first, the marker is directly incorporated into the intensity function; second, each marker can be regarded as an independent dimension to have a multi-dimensional temporal point process. In terms of the former approach, we still need to specify a proper form for the conditional intensity function. Moreover, due to the extra complexity of the function induced by the markers, people normally make strong assumptions that the marker is independent on the history [134], which greatly reduces the flexibility of the model. With respect to the latter method, it is very common to have large number of markers, which results in a sparsity problem associated with each dimension where only very few events can happen.

## 7.3 Recurrent Marked Temporal Point Process

Each parametric form of the conditional intensity function determines the temporal characteristics of a family of point processes. However, it will be hard to correctly decide which form to use without any sufficient prior knowledge in order to take into account both the marker and the timing information. To tackle this challenge, in this section, we propose a unified model capable of modeling a general nonlinear

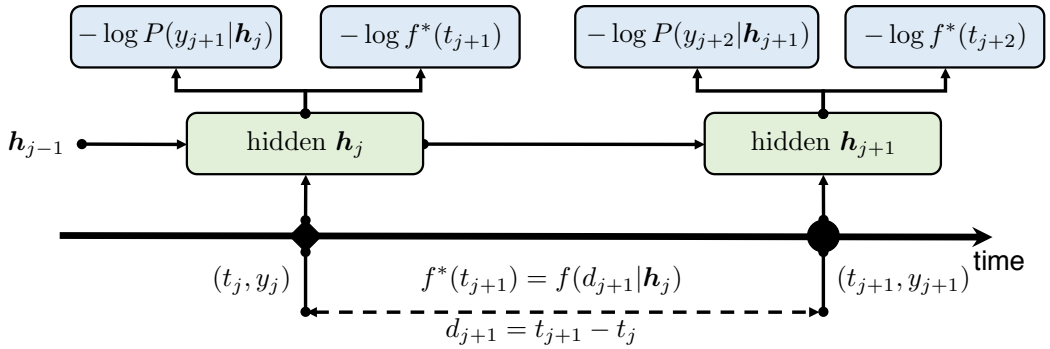



Figure 7.2: Illustration of Recurrent Marked Temporal Point Process. For each event with the timing $t_j$ and the marker $y_j$, we treat the pair $(t_j, y_j)$ as the input to a recurrent neural network unrolled to the $j$-th step where the hidden state $\boldsymbol{h}_j$ up to the time $t_j$ learns a general representation of a nonlinear dependency over both the timing and the marker information from past events. Note that the solid diamond and the circle indicates two events of different types $y_j \neq y_{j+1}$.

dependency over the history of both the event timing and the marker information.

### 7.3.1 Model Formulation

By carefully investigating the various forms of the conditional intensity function 7.2, 7.3, and 7.4, we can observe that they are inherently different representations and realizations of various kinds of dependency structures over the past events. Inspired by this critical insight, we seek to learn a general representation to *universally approximate* the unknown dependency structure over the history.

Recurrent Neural Network (RNN) is a feedforward neural network structure where additional edges, referred to as the recurrent edges, are added such that the outputs from the hidden units at the current time step are fed into them again as the future inputs at the next time step. As a consequence, the same feedforward neural network structure is replicated at each time step, and the recurrent edges connect the hidden units of the networks at adjacent time steps together along time, that is, the hidden nodes with recurrent edges not only receive the input from the current data sample but also from the hidden units in the last time step. This feedback mechanism creates an internal state of the network to memorize the influence of each past data sample.

In theory, finite-sized recurrent neural networks with sigmoidal activation units can simulate a universal Turing machine [147], which is able to perform an extremely rich family of computations. In practice, RNN has been shown to be a powerful tool for general purpose sequence modeling. For instance, in Natural Language Processing, recurrent neural network has state-of-the-arts predictive performance for sequence-to-sequence translations [77], image captioning [163], handwriting recognition [62] and even executing programs [174].

Our key idea is to let the RNN (or its modern variant LSTM [75], GRU [31], *etc.*) model the nonlinear dependency over both of the markers and the timings from past events. As shown in Figure 7.2, for the event occurring at the time $t_j$ of type $y_j$, the pair $(t_j, y_j)$ is fed as the input into a recurrent neural network unfolded up to the $j$-th event. The hidden state $\boldsymbol{h}_{j-1}$ represents the memory of the influence from the timings and the markers of past events. The neural network updates $\boldsymbol{h}_{j-1}$ to $\boldsymbol{h}_j$ by taking into account the effect of the current event $(t_j, y_j)$. Since now $\boldsymbol{h}_j$ represents the influence of the history up to the $j$-th event, the conditional density for the next event timing can be naturally represented as

$$f^*(t_{j+1}) = f(t_{j+1}|\mathcal{H}_t) = f(t_{j+1}|\boldsymbol{h}_j) = f(d_{j+1}|\boldsymbol{h}_j), \tag{7.5}$$

where $d_{j+1} = t_{j+1} - t_j$. As a consequence, we can depend on $\boldsymbol{h}_j$ to make predictions to the timing $\hat{t}_{j+1}$ and the type $\hat{y}_{j+1}$ of the next event.

The advantage of this formulation is that by the elegant relation 2.7, we are now able to capture a general form of the conditional intensity function $\lambda^*(t)$ without the need of specifying a fixed parametric specification for the dependency structure over the history. Figure 7.3 presents the overall architect of the proposed RMTPP. Given a sequence of events $\mathcal{S} = \big((t_j, y_j)_{j=1}^n\big)$, we design an RNN which computes a sequence of hidden units $\{\boldsymbol{h}_j\}$ by iterating the following components.

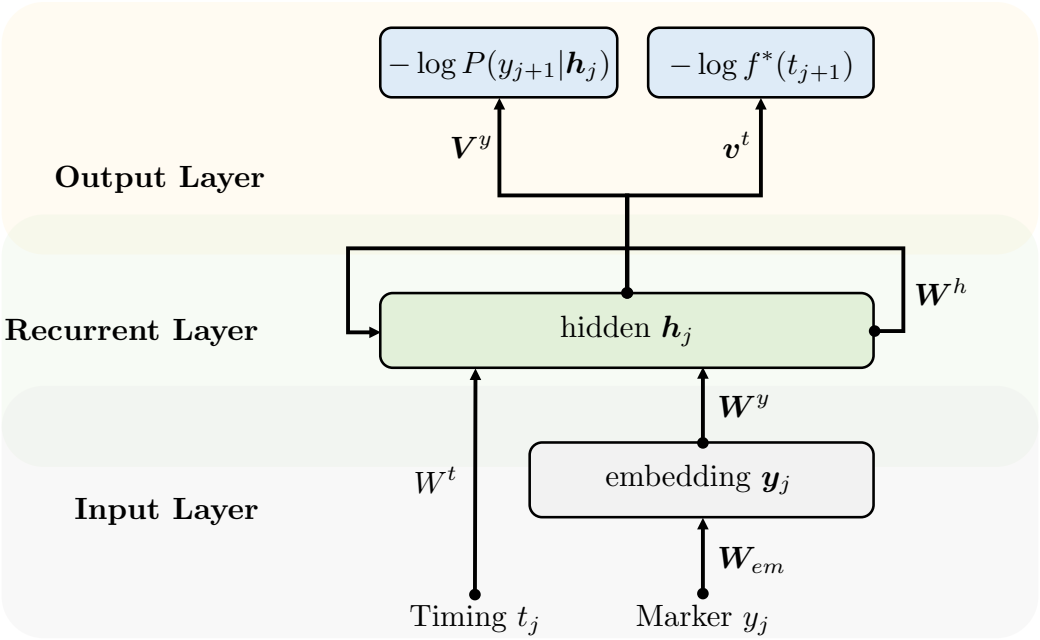



Figure 7.3: Architect of Recurrent Marked Temporal Point Process. For a given sequence $\mathcal{S} = \big((t_j, y_j)_{j=1}^n\big)$, at the $j$-th event, the marker $y_j$, represented by the one-hot encoding, is first embedded into a latent space. Then, the embedded vector and the temporal features extracted from the current time $t_j$ is fed into the recurrent layer. The recurrent layer learns a hidden representation that summaries the nonlinear dependency over the previous events. Based on the learned representation $\boldsymbol{h}_j$, it outputs the prediction for the next marker $\hat{y}_{j+1}$ and timing $\hat{t}_{j+1}$ to calculate the respective loss functions.

**Input Layer.** At the $j$-th event, the input layer first projects the sparse one-hot vector representation of the marker $y_j$ into a latent space. We add an embedding layer with the weight matrix $\boldsymbol{W}_{em}$ to achieve a more compact and efficient representation $\boldsymbol{y}_j = \boldsymbol{W}_{em}^\top y_j + \boldsymbol{b}_{em}$, where $\boldsymbol{b}_{em}$ is the bias. We learn $\boldsymbol{W}_{em}$ and $\boldsymbol{b}_{em}$ while we train the network. In addition, for the timing input $t_j$, we can extract the associated temporal features $\boldsymbol{t}_j$, such as the inter-event time $d_j = t_j - t_{j-1}$.

**Hidden Layer.** We update the hidden vector after receiving the current input and the memory $\boldsymbol{h}_{j-1}$ from the past. In RNN, we have:

$$\boldsymbol{h}_j = \max\left\{\boldsymbol{W}^y \boldsymbol{y}_j + \boldsymbol{W}^t \boldsymbol{t}_j + \boldsymbol{W}^h \boldsymbol{h}_{j-1} + \boldsymbol{b}_h, \boldsymbol{0}\right\}. \tag{7.6}$$

**Output Layer.** Given the learned representation $\boldsymbol{h}_j$, we model the marker generation as a multinomial distribution by

$$P(y_{j+1} = k|\boldsymbol{h}_j) = \frac{\exp\left(\boldsymbol{V}^y_{k,:}\boldsymbol{h}_j + b^y_k\right)}{\sum_{k=1}^{K}\exp\left(\boldsymbol{V}^y_{k,:}\boldsymbol{h}_j + b^y_k\right)}, \tag{7.7}$$

where $K$ is the number of markers and $\boldsymbol{V}^y_{k,:}$ is the $k$-th row of matrix $\boldsymbol{V}^y$.

Based on $\boldsymbol{h}_j$, we can now formulate the conditional intensity function by:

$$\lambda^*(t) = \exp\left(\underbrace{\boldsymbol{v}^{t^\top}\cdot\boldsymbol{h}_j}_{\text{history influence}} + \underbrace{w^t(t-t_j)}_{\text{current influence}} + \underbrace{b^t}_{\text{base intensity}}\right), \tag{7.8}$$

where $\boldsymbol{v}^t$ is a column vector, and $w^t$, $b^t$ are scalars.

**Model Justification** in steps:

- The first term $\boldsymbol{v}^{t^\top}\cdot\boldsymbol{h}_j$ represents the accumulative influence from the marker and the timing information of the past events. Compared to the fixed parametric formulations of (7.2), (7.3), and (7.4) for the past influence, we now have a highly non-linear general specification of the dependency over the history.
- The second term emphasizes the influence of the current event $j$.
- The last term gives a base intensity level for the occurrence of the next event.
- The exponential function outside acts as a non-linear transformation and guarantees that the intensity is positive.

In consequence, we formally define the Recurrent Marked Temporal Point Process as the following.

**Definition 7.1** *Recurrent Marked Temporal Point Process is a marked temporal point process with the conditional intensity function given by*

$$\lambda^*(t) = \exp\left(\boldsymbol{v}^{t^\top}\cdot\boldsymbol{h}_j + w^t(t-t_j) + b^t\right)$$

*where $t_j < t$, $\boldsymbol{h}_j$ is the hidden state of a recurrent neural network at the $j$-th event and $b^t$ is the base intensity.*

By invoking the elegant relation between the conditional intensity function and the conditional density function in (2.7), we can derive the likelihood that the next event will occur at time $t$ by

$$\begin{aligned} f^*(t) =& \lambda^*(t) \exp\left( - \int_{t_j}^{t} \lambda^*(\tau) d\tau \right) \\ =& \exp\left\{ \boldsymbol{v}^{t\top} \cdot \boldsymbol{h}_j + w^t(t - t_j) + b^t + \frac{1}{w} \exp(\boldsymbol{v}^{t\top} \cdot \boldsymbol{h}_j + b^t) \right. \\ & \left. - \frac{1}{w} \exp(\boldsymbol{v}^{t\top} \cdot \boldsymbol{h}_j + w^t(t - t_j) + b^t) \right\} \end{aligned} \tag{7.9}$$

Then, we can estimate the timing for the next event using the expectation

$$\hat{t}_{j+1} = \int_{t_j}^{\infty} t \cdot f^*(t) dt. \tag{7.10}$$

In general, the integration in (7.10) does not have analytic solutions, so we can apply commonly used numerical integration techniques [130] for one-dimensional functions to compute (7.10) instead.

Based on the hidden states of RNN, we are able to learn a unified representation of the dependency over the history. As a consequence, the direct formulation (7.8) of the conditional intensity function $\lambda^*(t_{j+1})$ captures both of the information from past event timings and event markers. On the other hand, since the prediction for the marker also depends nonlinearly on the past timing information, this may improve the performance of the classification task as well when both of these two information are correlated with each other. In fact, experiments on synthetic and real world datasets in the following experimental section do verify this mutual boosting claim.

### 7.3.2 Parameters Learning

Given a collection of sequences $\mathcal{C} = \{\mathcal{S}^i\}$, where $\mathcal{S}^i = \left((t_j^i, y_j^i)_{j=1}^{n_i}\right)$, we can learn the model by maximizing the joint log-likelihood of observing $\mathcal{C}$.

$$\ell\left(\{\mathcal{S}^i\}\right) = \sum_i \sum_j \left( \log P(y_{j+1}^i | \boldsymbol{h}_j) + \log f(d_{j+1}^i | \boldsymbol{h}_j) \right), \tag{7.11}$$

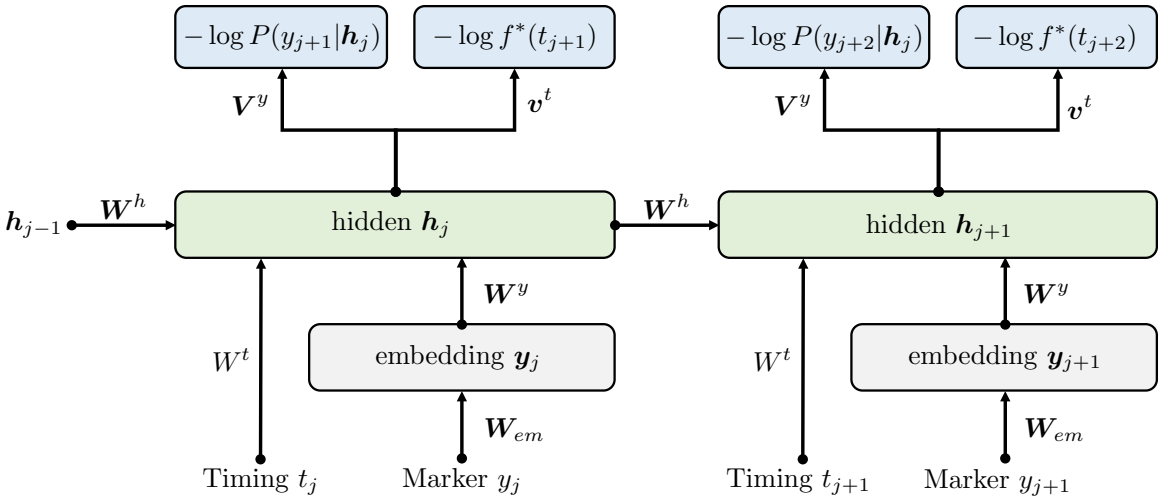



Figure 7.4: Illustration of training Recurrent Marked Temporal Point Process using Back Propagation Through Time (BPTT). Here we show an example of $b = 2$-step unrolling along time. The bias parameters are included in each layer by default without being shown here.

In order to utilize the sequential dependency information, we exploit the Back Propagation Through Time (BPTT) for training RMTPP. Given the size of BPTT as $b$, we can unroll our model in Figure 7.3 by $b$ steps, as shown in Figure 7.4.

In each training iteration, we take $b$ consecutive samples $\{(t_k^i, y_k^i)_{k=j}^{j+b}\}$ from a single sequence, apply the feed-forward operation through the network, and update the parameters with respect to the loss function. After we unroll the model for $b$ times along time, all the parameters are shared across these copies, and will be updated in a concurrent way described below.

We use the chain rule to derive the partial derivatives with respect to each parameter shown in Figure 7.4. Let $\boldsymbol{z}^y = \max\{\boldsymbol{V}^y\boldsymbol{h}_j + \boldsymbol{b}^y, 0\}$ and $z^t = \boldsymbol{v}^{t\top}\boldsymbol{h}_j + b^t$. $\boldsymbol{e}^y$ is the $K$-dimensional column vector where $K = |\mathcal{Y}|$ is the number of markers in total. Given the next marker $y_{j+1} = k$ in a sequence, we define the $k$-th component $\boldsymbol{e}_k^y = \partial\ell/\partial\boldsymbol{z}_k^y$ and $\boldsymbol{e}_i^y = 0$ for all the other components $i \neq k$. Similarly, we define $e^t = \partial\ell/\partial z^t$. Both $\boldsymbol{e}^y$ and $e^t$ can be treated as the 'error' with respect to the supervised information of the next event.

Then, we can propagate back these errors to the recurrent layers as below:

$$
\begin{aligned}
&\frac{\partial \ell}{\partial \boldsymbol{h}_j} = \boldsymbol{V}^{y\top}(\boldsymbol{e}^y \circ \mathbb{I}\{\boldsymbol{z}^y > 0\}) + \boldsymbol{v}^t e^t + \boldsymbol{W}^{h\top}\left(\frac{\partial \ell}{\partial \boldsymbol{h}_{j+1}} \circ \mathbb{I}\{\boldsymbol{h}_{j+1} > 0\}\right),\\
&\frac{\partial \ell}{\partial \boldsymbol{y}_j} = \boldsymbol{W}^{y\top}\left(\frac{\partial \ell}{\partial \boldsymbol{h}_j} \circ \mathbb{I}\{\boldsymbol{h}_j > 0\}\right),\\
&\frac{\partial \ell}{\partial \boldsymbol{V}^y} = \sum_{i=j}^{j+b} \boldsymbol{e}^{y,i}\left(\boldsymbol{h}_i \circ \mathbb{I}\{\boldsymbol{z}^y > 0\}\right)^\top, \quad \frac{\partial \ell}{\partial \boldsymbol{v}^t} = \sum_{i=j}^{j+b} \boldsymbol{h}_i e^t,\\
&\frac{\partial \ell}{\partial \boldsymbol{W}^h} = \sum_{i=j}^{j+b}\left(\frac{\partial \ell}{\partial \boldsymbol{h}_i} \circ \mathbb{I}\{\boldsymbol{h}_j > 0\}\right)\boldsymbol{h}_{i-1}^\top, \quad \frac{\partial \ell}{\partial \boldsymbol{W}^y} = \sum_{i=j}^{j+b}\left(\frac{\partial \ell}{\partial \boldsymbol{h}_j} \circ \mathbb{I}\{\boldsymbol{h}_j > 0\}\right)\boldsymbol{y}_i^\top\\
&\frac{\partial \ell}{\partial \boldsymbol{W}^t} = \sum_{i=j}^{j+b}\left(\frac{\partial \ell}{\partial \boldsymbol{h}_j} \circ \mathbb{I}\{\boldsymbol{h}_j > 0\}\right)\boldsymbol{t}_i^\top, \quad \frac{\partial \ell}{\partial \boldsymbol{W}_{em}} = \sum_{i=j}^{j+b}\frac{\partial \ell}{\partial \boldsymbol{y}_i}\boldsymbol{y}_i^{b\top}, \quad \frac{\partial \ell}{\partial \boldsymbol{b}_{em}} = \sum_{i=j}^{j+b}\boldsymbol{y}^i,\\
&\frac{\partial \ell}{\partial \boldsymbol{b}^y} = \sum_{i=j}^{j+b}\boldsymbol{e}^{y,i}, \quad \frac{\partial \ell}{\partial b^t} = \sum_{i=j}^{j+b} e^{t,i}, \quad \frac{\partial \ell}{\partial \boldsymbol{b}^h} = \sum_{i=j}^{j+b}\left(\frac{\partial \ell}{\partial \boldsymbol{h}_j} \circ \mathbb{I}\{\boldsymbol{h}_j > 0\}\right),
\end{aligned}
\tag{7.12}
$$

where $\circ$ is the element-wise operation, $\boldsymbol{e}^{y,i}$ is the vector $\boldsymbol{e}^y$ calculated from the $i$-th event, and $\boldsymbol{y}_i^b$ is the one-hot representation for the $i$-th marker $y_i$ in a sequence. For the case when mini-batch applies, we could simply average the derivatives achieved in each training sequence to reduce the variance of the gradient. The update rule for more complicated recurrent units, like LSTM [75] and GRU [31], can also be derived in similar way.

### 7.3.3 Efficient Implementation

In our algorithm framework, we need both sparse-format features (the marker $y_j$) and dense-format features (the time feature $t_j$) at time $t_j$. Meanwhile, the output is also mixed of discrete-value markers and real-value timings, which is further fed into different loss functions including the cross-entropy of the next marker prediction and the negative log-likelihood of the next event timing. As a result, we build an efficient and flexible platform particularly optimized for training general directed acyclic structured computational graph (DAG). The backend is supported via CUDA and MKL for GPU and CPU platform, respectively. Regarding the optimization

method, we simply used stochastic gradient descent (SGD) with mini-batch. We have utilized several techniques in training neural networks, including momentum [154] and regularizations. We have also tried the batch normalization [78], but since the network is not deep here, it does not contribute too much to our task.

## 7.4 Experiments

We evaluate our proposed Recurrent Marked Temporal Point Process in large-scale synthetic and real world data. We compare it to several discrete-time and continuous-time series models showing that our model is more robust to model misspecifications than these alternatives.

### 7.4.1 Baselines

To demonstrate the predictive performance of forecasting markers, we compare with the following discrete-time models:

- **Majority Prediction**. This is also known as the 0-order **M**arkov **C**hain (**MC-0**), where at each time step, we always predict the most popular marker regardless of the history. Most often, predicting the most popular type is a strong heuristic.

- **Markov Chain**. We compare with Markov models with varying orders from one to three, denoted as **MC-1**, **MC-2**, and **MC-3**, respectively.

To show the effectiveness of predicting time, we compare with the following continuous-time models:

- **ACD**. We fit a second-order autoregressive conditional duration process with the intensity function given in 7.4.

- **Homogeneous Poisson Process**. The intensity function is a constant, of which the inverse is equal to the average inter-event gaps.

- **Hawkes Process**. We fit a self-excitation Hawkes process with the intensity function given in 7.2, where the triggering kernel $\gamma(t, t') = \exp\left(-\frac{t-t'}{\sigma}\right), t > t'$.

- **Self-correcting Process**. We fit a self-correcting process with the intensity function given in 7.3.

Finally, we compare with the **C**ontinuous-**T**ime **M**arkov **C**hain (**CTMC**) model, which learns continuous transition rates between two states (or markers). This model predicts the next state with the earliest transition time, so it can predict both the marker and the timing for the next event jointly.

### 7.4.2 Synthetic Data

To show the robustness of RMTPP, we propose the following generative processes:

**Autoregressive Conditional Duration.** The conditional density function for the next duration $d_n = t_n - t_{n-1}$ conforms to an exponential distribution with the expectation determined by the past $m$ subsequent durations in the following form, which is denoted as ACD.

$$f(d_n|\mathcal{H}_{n-1}) = \alpha_n \exp(-\alpha_n \tau_n), \alpha_n = \left(\mu_0 + \gamma \sum_{i=1}^{m} d_{n-i}\right)^{-1}, \tag{7.13}$$

where $\mu_0$ is the base duration to generate the first event starting from zero, $d_1 \sim (\mu_0)^{-1} \exp(-d_1/\mu_0)$. We set $m = 2$, $\mu_0 = 0.5$ and $\gamma = 0.25$.

**Hawkes Process.** The conditional intensity function is given by $\lambda(t) = \lambda_0 + \alpha \sum_{t_i<t} \exp\left(-\frac{t-t_i}{\sigma}\right)$ where $\lambda_0 = 0.2$, $\alpha = 0.8$ and $\sigma = 1.0$. The Hawkes process is used to produce the self-excitation phenomena where recent events will trigger more events in the near future to produce clustered point patterns.

**Self-Correcting Process.** The conditional intensity function is given by $\lambda(t) = \exp\left(\mu t - \sum_{t_i<t} \alpha\right)$ where $\mu = 1$ and $\alpha = 0.2$. The self-correcting process illustrates

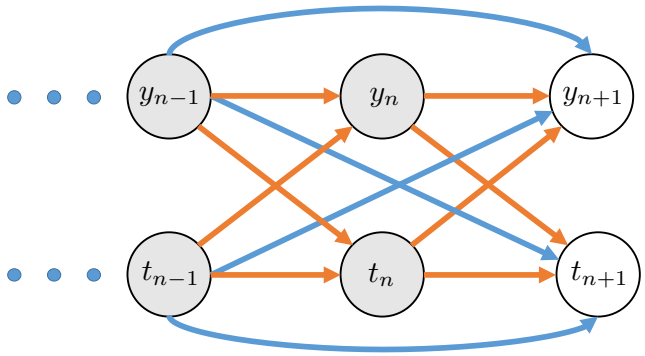



Figure 7.5: The next event type and timing variable pair $(t_{n+1}, y_{n+1})$ depends on the past two pairs $(t_n, y_n)$ and $(t_{n-1}, y_{n-1})$. Orange and blue lines denote the first-order and the second-order dependency, respectively.

the self-inhibiting phenomena where the occurrences of recent events will reduce the chance of new events in the near future to produce regular point patterns.

**State-Space Continuous-Time Model.** To model the influence from both markers and time, we further propose the State-Time Mixture model shown in Figure 7.5 with the following steps:

1. For each time $t_{n-1}$, we take the mod of $t_{n-1}$ by a period of $P = 24$. If the residual is greater than 12, the process is defined to be in the time state $r_{n-1} = 0$; otherwise, it is in the time state $r_{n-1} = 1$.

2. Based on the combination of both the time state $\{r_{n-j}\}_{j=1}^{m}$ and the marker state $\{y_{n-j}\}_{j=1}^{m}$ of the previous $m$ events, the process will have the marker $k$ for the next step in the probability $P(y_n = k | \{y_{n-j}\}_{j=1}^{m}, \{r_{n-j}\}_{j=1}^{m})$.

3. Similarly, based on the combination of $\{y_{n-j}\}_{j=1}^{m}$ and $\{r_{n-j}\}_{j=1}^{m}$ from the previous $m$ events, the duration $d_n = t_n - t_{n-1}$ has a Poisson distribution with the expectation determined by $\{r_{n-j}\}_{j=1}^{m}$ and $\{y_{n-j}\}_{j=1}^{m}$ jointly. Here, we use the Poisson distribution to mimic the elapsed time units (*e.g.*, hours, minutes).

In our experiments, without loss of generality, we set the total number of markers to two, $m = 3$ and randomly initialize the transition probabilities between states.

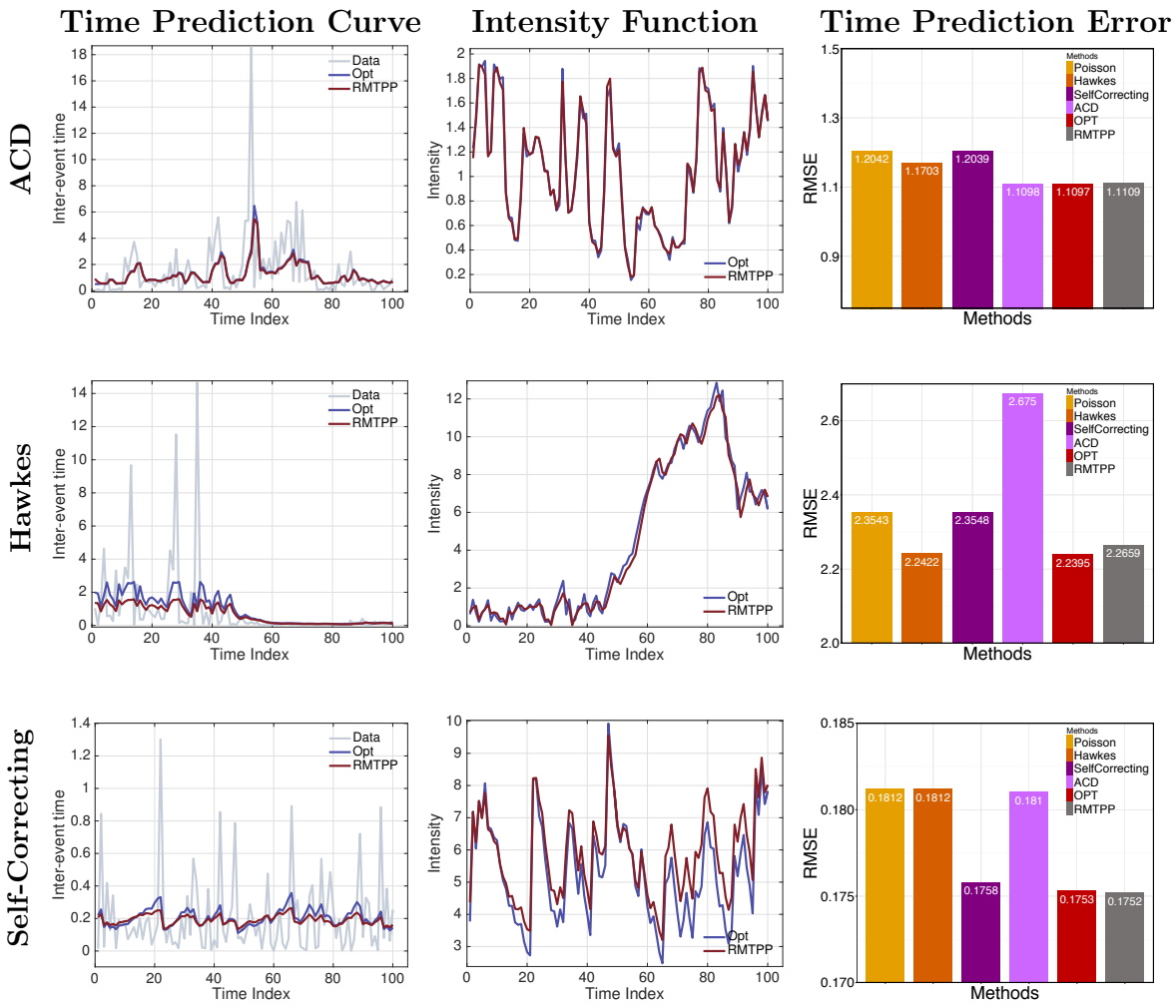
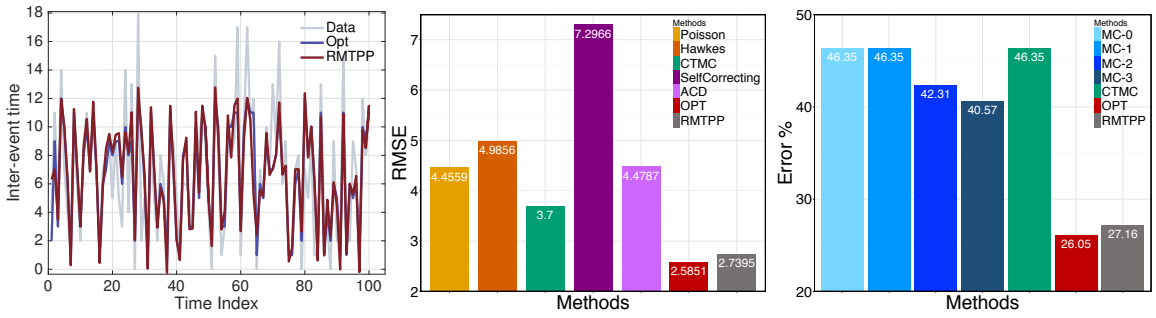



Figure 7.6: Inter-event time predictions on the testing time-series data produced from different processes. Left column is the predicted inter-event time. Blue curve is the optimal estimator which knows the true functional form for the conditional density function with the true parameters and predict the inter-event time with the expectation. Red curve is the prediction given by RMTPP without any prior knowledge about each specific form instead. Middle column shows the learned intensity functions vs. the respective true ones. Right column gives the overall testing RMSE of predicting the timings from different processes.

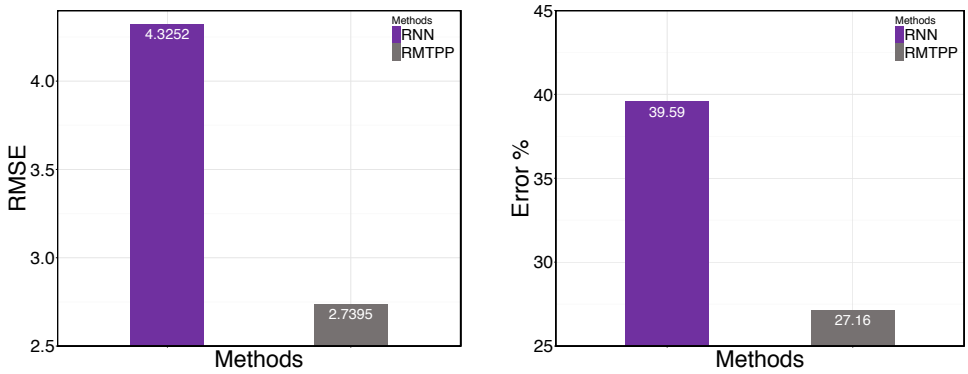
**Experimental Results.** Figure 7.6 presents the predictive performance of RMTPP fitted to different types of time-series data. In each case of Figure 7.6, we simulate 1,000,000 events, use 90% for training and the rest 10% for testing. We first compare the predictive performance of our model with the optimal estimator in the left column of Figure 7.6. The optimal estimator is given the true functional form of the dependency structure on the past events and the respective model parameters. We treat the expectation of the time interval between the current and the next event as our estimation. In the left column of Figure 7.6, grey curves are the observed inter-event durations from 100 successive events in the testing data. Blue curves are the respective expectations given by the optimal estimator. Red curves are the predictions from our RMTPP model. We can observe that even though RMTPP has no prior knowledge about the true functional form of each process, its predictive performance is almost consistent with the respective optimal estimator.

The middle column of Figure 7.6 compares the learned conditional intensity functions (red curves) with the true ones (blue curves). It clearly demonstrates that RMTPP is able to adaptively and accurately capture the unknown heterogeneous temporal dynamics of different time-series data. In particular, because the order of dependency over the history is fixed, RMTPP almost exactly learns the conditional intensity function of the autoregressive conditional duration (ACD) process with comparable BPTT steps expanded in time. The Hawkes and the self-correcting processes are more challenging in that the conditional intensity function depends on the whole history. Because the events are far from being uniformly distributed, the influence from individual past event to the occurrence of new future events can vary widely. From this perspective, these processes essentially have random varying order dependency on the history compared to the fixed order dependency of ACD. However, we can observe that with properly chosen BPTT steps, RMTPP can accurately capture the general shape and each single change time point of the true intensity function.




(a) Time Prediction Curve (b) Time Prediction Error (c) Marker Prediction Error

Figure 7.7: Performance evaluation of predicting timings and markers on the state-space continuous-time model.




(a) Time Prediction Error (b) Marker Prediction Error

Figure 7.8: Predictive performance comparison with RNN which is trained for predicting the next timing only in (a), and for predicting the next marker only in (b).

Especially for the Hawkes process, the abruptly increased intensity from time index 60 to 100 results in 40 events in a very tiny time interval, which can be seen in the second panel of the left column of Figure 7.6 where the respective inter-event time is almost close to zero. But still, the predictions of RMTPP can capture the trend of the true data. The right column of Figure 7.6 reports the overall RMSE of different processes between the predictions and the true testing data. We can observe that RMTPP has very strong competitive performance and better robustness against model misspecification to capture the heterogeneity of the latent temporal dynamics of different time-series data compared to other parametric alternatives.

In addition to time, the state-space continuous-time model also includes the

marker information. Figure 7.7 compares the error rates of different processes in predicting both timings and markers. Compared to the other alternatives, the performance of RMTPP for predicting both the timing and the marker for the future event is again consistent with that of the optimal estimator without knowing any prior knowledge about the true functional form of the underlying generative process.

Finally, since the marker and the timing of future events depend on the past marker and timing information jointly, we would like to further investigate whether learning a unified representation of the joint information can improve the prediction of each task (label and timing) individually. As a consequence, we train an RNN by only using the temporal and the marker information, respectively. Figure 7.8 gives the comparison results between RMTPP and RNN where in panel (a), RNN has the 4.3522 RMSE while RMTPP achieves a 2.7395 RMSE, and in panel (b), RNN reports 39.59% classification error while RMTPP reaches to the 27.16% level. Clearly, both cases together verify that jointly model the information from both markers and timings can boost the prediction performance for future events.

### 7.4.3 Real Data

We evaluate the predictive performance of RMTPP on real world datasets from a diverse range of domains described below.

**New York City Taxi Dataset.** The NYC taxi dataset[1] contains ∼173 million trip records of individual Taxi for consecutive 12 months in 2013. The location information is available in the form of latitude/longitude coordinates. Each record also contains the temporal information of pick-up (drop-off) passengers associated with every trip. We have used NYC Neighborhood Names GIS dataset[2] to map the coordinates to neighborhood names. For those coordinates of which the location name

[1] http://www.andresmh.com/nyctaxitrips/

[2] https://data.cityofnewyork.us/City-Government/Neighborhood-Names-GIS/99bc-9p23

is not directly available in the GIS dataset, we use geodesic distance to map them to the nearest neighborhood name. With this process we obtained 299 unique locations as our markers. Then the dataset was distilled to produce temporal sequences of events associated with each taxi. An event in this case is a pickup record for a taxi. Further, we have divided each single sequence of a taxi into multiple fine-grained subsequences associated with the same taxi with roughly regular point patterns so that no two consecutive events within a subsequence can have a huge gap more than 12 hours. We obtained 670,753 sequences in total out of which 536,603 were used for training and 134,150 were used for testing purpose. The minimum sequence length (no. of events in a sequence) is 20 and the maximum is 12,568 with an average length of 245. Prediction task was performed for both location and time of the next pickup event. Figure 7.9 gives sample sequences of pickup events for 10 taxis. All the points (pickup locations) with the same color belong to the same taxi. It can be seen that taxi 1 picks up passengers only from Astoria and Bedford-Stuyvesant neighborhoods. Similarly, taxi 2 picks up passengers only between Bushwick and Cobble Hill while taxi 9 and 10 have varying pickup locations around Manhattan and Midtown area.

**Financial Transaction Dataset.** We have collected a raw limited order book data from NYSE of the high-frequency transactions for a stock in one day. It contains 0.7 million transaction records, each of which records the time (in millisecond) and the possible action (B = buy, S = sell). We treat the type of actions as markers. The input data is a single long sequence with 624,149 events for training and 69,350 events for testing. The task is to predict which action will be taken next at what time.

**Electrical Medical Records.** MIMIC II medical dataset is a collection of de-identified clinical visit records of Intensive Care Unit patients for seven years. We have filtered out 650 patients and 204 diseases. Each event records the time when a patient has a visit to the hospital. A patient could receive more than one diagnosis

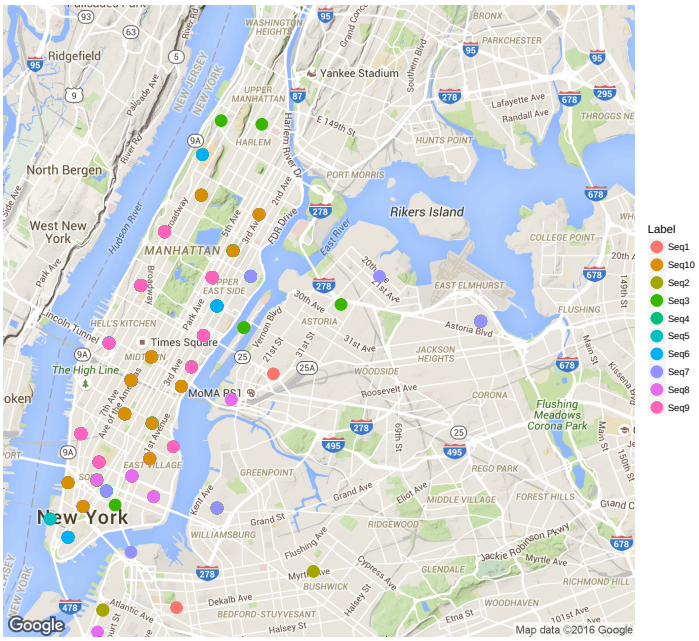



Figure 7.9: Sample sequences of pickup events in New York City Taxi Data.

at a single visit, one of which is assigned as the primary diagnosis. We have used the sequences of 585 patients to train, and the rest for test. The goal is to predict which major disease will happen to a given patient at what time in the future.

**Stack OverFlow Dataset.** Stack Overflow is a well-known question-answering website about programming[3]. It exploits badges and reputation as incentives to encourage user engagement and guide behaviors [61]. It makes all the data about its website available periodically for download[4]. Since the inception of the website in 2008 till 2014-9-15, Stack Overflow has awarded ∼10 million badges to ∼1.7 million users. There are 81 types of badges non-topical (i.e., non-tag affiliated) badges which can be awarded either only once (e.g. Altruist, Inquisitive, etc.) or multiple times (e.g. Stellar Question, Guru, Great Answer, etc.) to a user. We ignore the badges which can be awarded only once. The badges which can be given to users multiple times are awarded asynchronously and are usually triggered by the actions of other

[3]https://stackoverflow.com
[4]https://archive.org/details/stackexchange

users on the site (e.g., after a fixed number of votes on a question or an answer). We first select users who have earned at least 40 badges between 2012-01-01 and 2014-01-01 and then those badges which have been awarded at least 100 times to the users selected in the first step. We have removed such users who have been instantaneously awarded multiple badges due to technical issues on Stack Overflow servers. In the end, we have $\sim$6 thousand users with a total of $\sim$480 thousand events where each badge is treated as a marker.

**Experimental Results.** We compare and report the predictive performance of different models on the testing data of each dataset in Figure 7.10. The hyper-parameters of RMTPP across all these datasets are tuned as following: learning rate in $\{0.1, 0.01, 0.001\}$; hidden layer size in $\{64, 128, 256, 512, 1024\}$; momentum = 0.9 and L2 penalty = 0.001; and batch-size in $\{16, 32, 64\}$.

Figure 7.10 compares the predictive performance of forecasting markers and timings for the next event of different processes across the four real datasets. RMTPP outperforms the other alternatives with lower errors for predicting both timings and markers. Because the MIMIC-II dataset has many short sequences and is the smallest out of the four datasets, increasing the order of Markov chain will decrease its classification performance, and the Hawkes process also slightly performs better in predicting to the timings.

We also compare RMTPP with RNN trained only with the marker and with the timing information separately in Figure 7.11. We can observe that RMTPP trained by incorporating both the past marker and timing information performs consistently better than RNN trained with either one source of the information alone, which further verifies in practice that the joint marker and timing information does help to boost the prediction performance. Finally, Figure 7.12 further shows the empirical

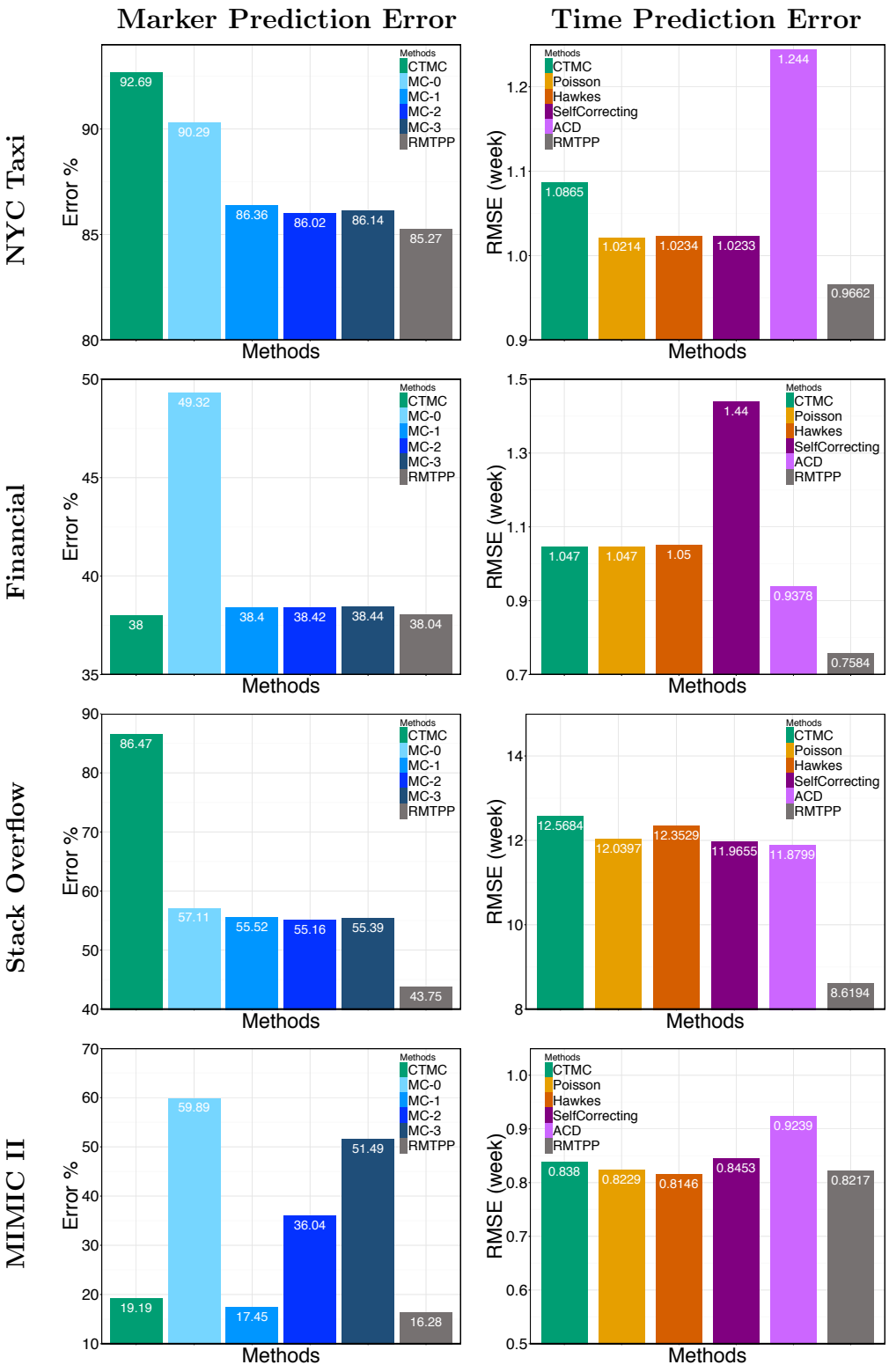



Figure 7.10: Performance evaluation for predicting both marker and timing of the next event. The left column presents the classification error of predicting markers, and the right column gives the RMSE of predicting the timings.

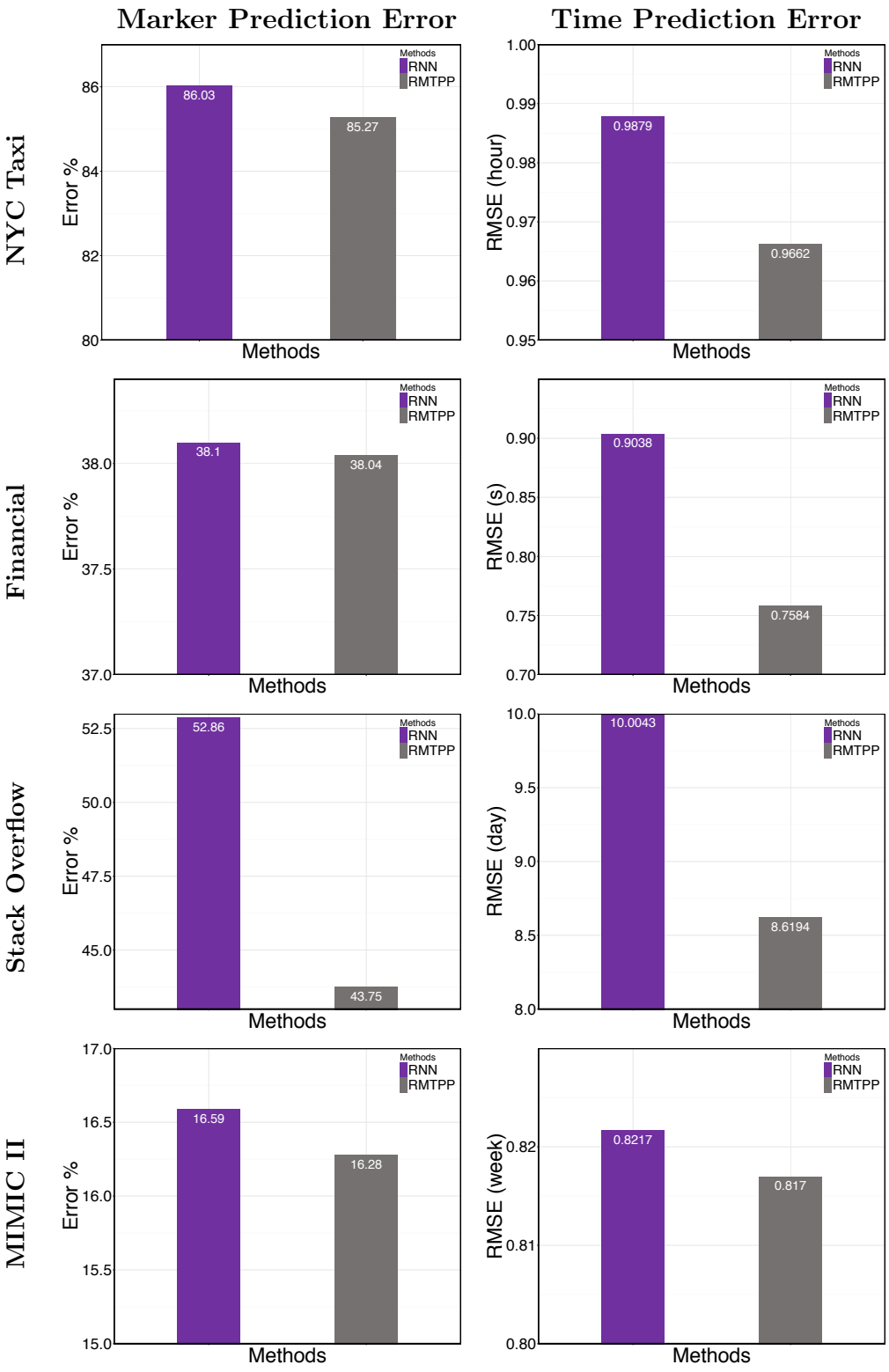
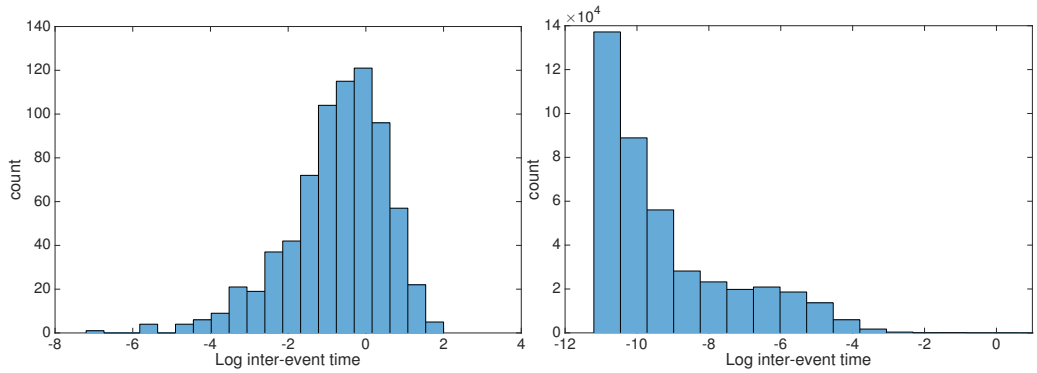



Figure 7.11: Predictive performance comparison with RNN which are trained only using the markers in the left column and only using the temporal information in the right column.




(a) Stack Overflow (b) Financial Transaction

Figure 7.12: Empirical distribution for the inter-event times. The x-axis is in log-scale.

distribution for the inter-event times on the Stack Overflow and the financial transaction data. Compared to the other temporal processes of fixed parametric forms, even though the real datasets might have quite different characteristics, our Recurrent Marked Temporal Point Process is still able to to produce better performance for both tasks in general.

## 7.5 Summary

In this chapter, we present the Recurrent Marked Temporal Point Process. Essentially, RMTPP builds a connection between recurrent neural networks and point processes. The recurrent neural network can have different architects, including the classic RNN and modern LSTM, GRU, *etc.* Besides, in addition to the inter-event temporal features, our model can be generalized to incorporate other contextual information. For instance, given a collection of sequences, each of which corresponds to the activity trace of a single user, in addition to training a global model, we can also take the extra user-profile features into account for personalization. Furthermore, based on the structural information of social networks, our model can be generalized such that the prediction of one user sequence not only depends on her own history but also depends on the other nodes' history to capture their interactions.

To conclude, RMTPP captures time-series data which have inherent general non-linear dependence over the history. It inherits the advantages from both the recurrent neural networks and the temporal point processes to predict both the marker and the timing for the next event without any prior knowledge about the hidden functional forms of the latent temporal dynamics. Experiments on both synthetic and real world datasets demonstrate that RMTPP is robust to model mis-specifications and has consistently better performance compared to the other alternatives.

In addition to markers, for many user-generated data, like tweets, documents, news articles, *etc.*, rich content information is also an important source for us to organize and extract meaningful insights and knowledge about the subjects being reported and studied. In the next chapter, we will illustrate the flexibility of our framework in combining the temporal information with other type of data so as to extract more comprehensive knowledge than using any of them alone.

# CHAPTER VIII

# COMBING TEMPORAL AND TEXTUAL DATA

Real-world data tend to have clustering structures which arise not only because of the similarity in content but also due to the closeness in time. Clusters in document streams, such as online news articles, can be induced by their textual contents, as well as by the temporal dynamics of their arriving patterns. Can we leverage both sources of information to obtain a better clustering of the documents, and distill information that is not possible to extract using contents only?

In this chapter, we propose a novel random process, referred to as the Dirichlet Point Process (DPP), to take into account both information in a unified model. A distinctive feature of the proposed model is that the preferential attachment of items to clusters according to cluster sizes, originally presented in Dirichlet processes, is now driven by the intensities of cluster-wise temporal point processes. This new process establishes a previously unexplored connection between Bayesian Nonparametrics and temporal point processes, which makes the number of clusters (dimensions) grow to accommodate the increasing complexity of online streaming contents, and at the same time, adapts to the ever changing dynamics of the respective continuous arrival timings. We have conducted large-scale experiments on both synthetic and real world news articles, and show that Dirichlet point processes can recover both meaningful topics and temporal dynamics, which leads to better predictive performance in terms of the arrival time of future documents.

## 8.1 Introduction

Online news articles, blogs and tweets tend to form clusters around real life events and stories on certain topics [3, 4, 37, 153, 21, 165]. Such data are generated by myriads

of online media sites in real-time and in large volumes. It is a critically important task to effectively organize these articles according to their contents such that online users can quickly sift and digest them.

Besides textual information, temporal information also provides very good clues on the clustering of online document streams. For instance, weather reports, forecasts and warnings of the blizzard in New York city this year appeared online even before the snowstorm actually started[1]. As the blizzard conditions gradually intensified, more subsequent blogs, posts and tweets were triggered around this event in various online social media. Such self-excitation phenomenon often leads to many closely related articles within a short period of time. Later, after the influence of the event past its peak, *e.g.*, the blizzard eventually stopped, public attention gradually turned to other events, and the following articles on the blizzard faded out eventually.

Furthermore, depending on the nature of real life events, relevant news articles can exhibit very different temporal dynamics. For instance, articles on emergency or incidents may rise and fall quickly, while some other stories, gossips and rumors may have a far reaching influence, *e.g.*, related posts about a Hollywood blockbuster can continue to appear as more details and trailers are revealed. As a consequence, the clustering of document streams can be improved by taking into account the underlying heterogeneous temporal dynamics. Distinctive temporal dynamics will also help us to disambiguate different clusters of similar topics emerging closely in time, to track their popularity and to predict the future trends.

Such problem of modeling time-dependent topic-clusters has been attempted by [3, 4], where the Recurrent Chinese Restaurant Process(RCRP) [6] has been proposed to model each topic-cluster of a news stream. However, one of the main deficiencies of the RCRP and related models [37] is that they require an explicit division of the event

[1] http://www.usatoday.com/story/weather/2015/01/25/northeast-possibly-historic-blizzard/22310869/

stream into unit episodes. Although this was ameliorated in the DD-CRP model [22] simply by defining a continuous weighting function, it does not address the issue that the actual *counts* of events are nonuniform over time. Artificially discretizing the time line into bins introduces additional tuning parameters, which are not easy to choose optimally. Therefore, in this chapter, we propose a novel random process, referred to as the Dirichlet Point Process (DPP), to take into account both sources of information to cluster *continuous-time* document streams.

The rest of this chapter is then organized as follows: Section 8.2 reviews some basic concepts from Bayesian Nonparametrics. In Section 8.3 we present our proposed Dirichlet Point process. In Section 8.4 we apply the Dirichlet point process to model document streams. In Section 8.5 we develop an efficient online inference algorithm which can scale up to millions of news articles with near constant processing time per document and moderate memory consumption. In Section 8.6, we conduct large-scale experiments on both synthetic and real-world datasets to evaluate the performance of DPP. Finally, we summarize the major contributions in Section 8.7.

## 8.2 Bayesian Nonparametrics

**The Dirichlet Process (DP) [8]** is one of the most basic Bayesian nonparametric processes, parameterized by a concentration parameter $\alpha > 0$ and a base distribution $G_0(\theta)$ over a given space $\theta \in \Theta$. A sample $G \sim DP(\alpha, G_0)$ drawn from a DP is a discrete distribution by itself, even the base distribution is continuous. Furthermore, the expected value of $G$ is the base distribution, and the concentration parameter controls the level of discretization in $G$: in the limit of $\alpha \to 0$, a sampled $G$ is concentrated on a single value, while in the limit of $\alpha \to \infty$, a sampled $G$ becomes continuous. In between are the discrete distributions with less concentration as $\alpha$ increases.

Since $G$ itself is a distribution, we can draw samples $\theta_{1:n}$ from it, and use these

Table 8.1: Table of symbols

| Symbol | Description |
|---|---|
| $DP(\alpha, G_0)$ | Dirichlet process with concentration parameter $\alpha$ and mean distribution $G_0$ |
| $\theta_k$ | Parameter of event type |
| $\theta_0$, $\alpha_0$ | Hyper-parameter for the Dirichlet distribution |
| $\theta_{1:n}$ | Collection of $\{\theta_1, \ldots, \theta_{n-1}\}$ |
| $\{\theta_k\}$ | A set of distinct values in $\theta_{1:n}$ |
| $\delta(\cdot)$ | Indicator function |
| $\gamma_\theta(\cdot, \cdot)$ | Triggering kernel associated with event type of parameter $\theta$ |
| $\lambda_\theta(t)$ | Intensity function associated with event type of parameter $\theta$ |
| $\lambda_0$ | The intensity of the background homogeneous Poisson process for generating new clusters |
| Poisson$(g(t))$ | Poisson process with the intensity $g(t)$ |
| Multi$(\theta)$ | Multinomial distribution with parameter $\theta$ |
| $s_n$ | Cluster index variable for the $n$-th document |
| $s_n^f$ | Cluster index variable for the $n$-th document via particle $f$ |
| $d_n$ | The $n$-th document |
| $d_{1:n}$, $t_{1:n}$, $s_{1:n}$ | Collection of $n$ documents, their arrival time and cluster indices |
| $f$ | The $f$-th particle |
| $w_n^f$ | The weight of particle $f$ when the $n$-th document arrives |
| $V$ | Total vocabulary size |
| $C^{d_n}$ | Word count in document $n$ |
| $C^{s_n \backslash d_n}$ | Word count of cluster $s_n$ excluding the $n$-th document |
| $C_v^{d_n}$, $C_v^{s_n \backslash d_n}$ | Count of the $v$-th word in document $n$ and the cluster $s_n$ excluding the $n$-th document |
| $\boldsymbol{\alpha}_{\theta_{s_n}}$, $\boldsymbol{\alpha}_{s_n}$ | Triggering kernel parameters for cluster $s_n$ |

samples as the parameters for models of clusters. Equivalently, let $\theta_{1:n}$ denote the collection of $\{\theta_1, \ldots, \theta_n\}$, and $\{\theta_k\}$ be the set of distinct values in $\theta_{1:n}$. Instead of first drawing $G$ and then sampling $\theta_{1:n}$, this two-stage process can be simulated as follows:

1. Draw $\theta_1$ from $G_0$.

2. For $n > 1$:

   (a) With probability $\frac{\alpha}{\alpha+n-1}$ draw $\theta_n$ from $G_0$.

   (b) With probability $\frac{m_k}{\alpha+n-1}$ reuse $\theta_k$ for $\theta_n$, where $m_k$ is the number of previous samples with value $\theta_k$.

This simulation process is also called Chinese Restaurant Process(CRP), which captures "the rich get richer" or preferential attachment phenomenon. Essentially, in this CRP metaphor, a Chinese restaurant has an infinite number of tables (each corresponding to a cluster). The $n$th customer $\theta_n$ can either choose a table with $m_k$ existing customers with probability $\frac{m_k}{n-1+\alpha}$, or start a new table with probability $\frac{\alpha}{n-1+\alpha}$. Formally, the conditional distribution of the $\theta_n$ can be written as a mixture:

$$\theta_n \,|\, \theta_{1:n-1} \sim \sum_k \frac{m_k}{n-1+\alpha}\delta(\theta_k) + \frac{\alpha}{n-1+\alpha}G_0(\theta). \tag{8.1}$$

In other words, it is more likely to sample from larger clusters, and the probability is proportional to the size of that cluster. Since the model allows new clusters to be created with a small probability, the model has the potential to generate infinite number of clusters adapted to the increasing complexity of the data. Thus the Dirichlet process is often used as a prior for the parameters of clustering models.

**The Recurrent Chinese Restaurant Process (RCRP)** is an extension of the DP which takes into account the temporal coherence of clusters for streaming documents divided into episodes [6]. One can think of RCRP as a discrete-time sequence of DPs, one for the data in each episode. The clusters in these DPs are shared, and the DPs appearing later in time can have a small probability to create new clusters.

More specifically, the conditional distribution of the $n$th value, $\theta_{t,n}$, sampled in

episode $t$ can be written as a mixture

$$\theta_{t,n} \,|\, \theta_{1:t-1,:},\, \theta_{t,1:n-1} \sim \sum_k \frac{m_{k,t} + m'_{k,t}}{\sum_j (m_{j,t} + m'_{j,t}) + \alpha} \delta(\theta_k) + \frac{\alpha}{\sum_j (m_{j,t} + m'_{j,t}) + \alpha} G_0, \tag{8.2}$$

where $\theta_{1:t-1,:}$ is the set of all samples drawn in previous episodes from 1 to $t-1$, and $\theta_{t,1:n-1}$ is the set of samples drawn in the current episode $t$ before $\theta_{t,n}$. The statistic $m_{k,t}$ is the number of previous samples in episode $t$ with value $\theta_k$, and $m'_{k,t}$ captures related information in $\theta_{1:t-1,:}$ about the value $\theta_k$. The latter quantity, $m'_{k,t}$, can be modeled in many ways. For instance, the original model in [6] applies a Markov Chain to model $m'_{k,t}$, and later follow-ups [3, 4] use a weighted combination of counts from recent $\Delta$ episodes

$$m'_{k,t} = \sum_{j=1}^{\Delta} e^{-\frac{j}{\beta}} m_{k,t-j}, \tag{8.3}$$

with an exponential kernel parametrized by the decaying factor $\beta$. Essentially, it models the decaying influence of counts from previous episodes across time. RCRP can also be used as a prior for the parameters of clustering models. For instance, Figure 8.2(a) shows a combination of RCRP and a bag-of-words model for each cluster.

However, RCRP requires artificially discretizing the time line into episodes, which is unnatural for continuous-time online document streams. Second, different type of clusters is likely to occupy very different time scales, and it is not clear how to choose the time window for each episode in advance. Third, the temporal dependence of clusters across episodes is hard-coded in (8.3), and it is the same for different clusters. Such design cannot capture the distinctive temporal dynamics of different type of clusters, such as related articles about disasters vs. Hollywood blockbusters, and fail to learn such dynamics from real data. We will use the temporal point process to address these drawbacks of RCRP when handling temporal dynamics.

## 8.3 Dirichlet Point Process

The key idea of Dirichlet Point Process is to have the Temporal Point Process model the rate ***intensity*** of events (*e.g.*, the arrivals of documents), while the Dirichlet Process captures the ***diversity*** of event types (*e.g.*, clusters of documents). Formally, we define the Dirichlet Point Process in below.

**Definition 8.1** *The Dirichlet Point Process is an infinite-dimensional temporal point process, which consists of a homogeneous Poisson process $\gamma_0$ and a potentially infinite number of temporal point processes $\{\gamma_d(t)\}_{d=1}^{n}$. With the probability $\frac{\gamma_d(t)}{\sum_{k=1}^{n}\gamma_k(t)+\gamma_0}$, an event at time $t$ will be attributed to one of the existing dimensions $\{1,\ldots,n\}$; with the probability $\frac{\gamma_0}{\sum_{k=1}^{n}\gamma_k(t)+\gamma_0}$, an event at time $t$ will be attributed to a newly generated dimension with the conditional intensity $\gamma_{n+1}(t)$.*

Because Hawkes process [72] is able to well capture the dependency over history and the clustering event patterns over time, without loss of generality, we use the Hawkes process [72] to describe the evolutionary process of each cluster (dimension). That is, we explicitly treat each cluster as one dimension of a multivariate Hawkes process. The intensity function of the Hawkes process on each dimension can be expressed as

$$\lambda(t) = \gamma_0 + \alpha \sum_{t_i \in \mathcal{T}} \gamma(t, t_i) \tag{8.4}$$

where $\gamma(t, t_i) \geqslant 0$ is the triggering kernel capturing temporal dependencies, $\gamma_0 \geqslant 0$ is a baseline intensity independent of the history and the summation of kernel terms is history dependent and a stochastic process by itself. The kernel function can be chosen in advance, *e.g.*, $\gamma(t, t_i) = \exp(-|t - t_i|)$ or $\gamma(t, t_i) = \delta(t > t_i)$, or directly learned from data. We also use the notation Poisson$(g(t))$ to denote a Poisson process [88] where $g(t)$ is the respective intensity function.

To begin with, DPP is parametrized by an intensity parameter $\lambda_0 > 0$, a base distribution $G_0(\theta)$ over a given space $\theta \in \Theta$ and a collection of triggering kernel

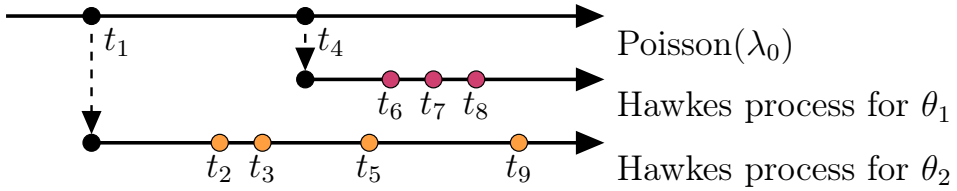



Figure 8.1: An illustration of Dirichlet point process. A background Poisson process with intensity $\lambda_0$ sampled the starting time points $t_1$ and $t_4$ for two different event types with the respective parameter $\theta_1$ and $\theta_2$. These two initial events then generate a Hawkes process of their own, with events at time $\{t_2, t_3, t_5, t_9\}$ and $\{t_6, t_7, t_8\}$, respectively.

functions $\{\gamma_\theta(t, t')\}$ associated with each event type of parameter $\theta$. Then, we can generate a sequence of samples $\{(t_i, \theta_i)\}$ as follows:

1. Draw $t_1$ from Poisson($\lambda_0$) and $\theta_1$ from $G_0(\theta)$.
2. For $n > 1$:
   (a) Draw $t_n > t_{n-1}$ from Poisson$\big(\lambda_0 + \sum_{i=1}^{n-1} \gamma_{\theta_i}(t, t_i)\big)$.
   (b) Draw $\theta_n$ from $G_0(\theta)$ with probability:
   $$\frac{\lambda_0}{\lambda_0 + \sum_{i=1}^{n-1} \gamma_{\theta_i}(t_n, t_i)}.$$
   (c) Reuse $\theta_k$ for $\theta_n$ with probability:
   $$\frac{\lambda_{\theta_k}(t_n)}{\lambda_0 + \sum_{i=1}^{n-1} \gamma_{\theta_i}(t_n, t_i)},$$
   where $\lambda_{\theta_k}(t_n) := \sum_{i=1}^{n-1} \gamma_{\theta_i}(t_n, t_i)\delta(\theta_i = \theta_k)$ is the intensity of a Hawkes process for previous events with value $\theta_k$.

Figure 8.1 gives an intuitive illustration of the Dirichlet Point process. Compared to the Dirichlet process, the intensity parameter $\lambda_0$ here serves the similar role to the concentration parameter $\alpha$ in the Dirichlet process. Instead of counting the number, $m_k$, of samples within a cluster, the Dirichlet Point process uses the intensity function

$\lambda_{\theta_k}(t)$ of a Hawkes process which can be considered as a temporally weighted count. For instance, if $\gamma_\theta(t, t_i) = \mathbb{I}[t > t_i]$, then $\lambda_{\theta_k}(t) = \sum_{i=1}^{n-1} \mathbb{I}[t > t_i]\mathbb{I}[\theta_i = \theta_k]$ is equal to $m_k$. If $\gamma_\theta(t, t_i) = \exp(-|t - t_i|)$, then $\lambda_{\theta_k}(t) = \sum_{i=1}^{n-1} \exp(-|t - t_i|)\mathbb{I}[\theta_i = \theta_k]$, and each previous event in the same cluster contributes a temporally decaying increment. Other triggering kernels associated with $\theta_i$ can also be used or learned from data. Thus the Dirichlet Point process is more general than the Dirichlet process and can generate both preferential-attachment type of clustering and rich temporal dynamics.

From the view of a temporal point process, the generation of the event timing in Dirichlet point process can also be viewed as the superposition of a Poisson process $\lambda_0$ and several Hawkes processes (conditional Poisson processes), one for each distinctive value of $\theta_d$ and with intensity $\lambda_{\theta_d}(t)$. Thus the overall event intensity is the sum of the intensities from individual processes [88]

$$\bar{\lambda}(t) = \lambda_0 + \sum_{d=1}^{D} \lambda_{\theta_d}(t),$$

where $D$ is the total number of distinctive values $\{\theta_i\}$ in the DPP up to time $t$.

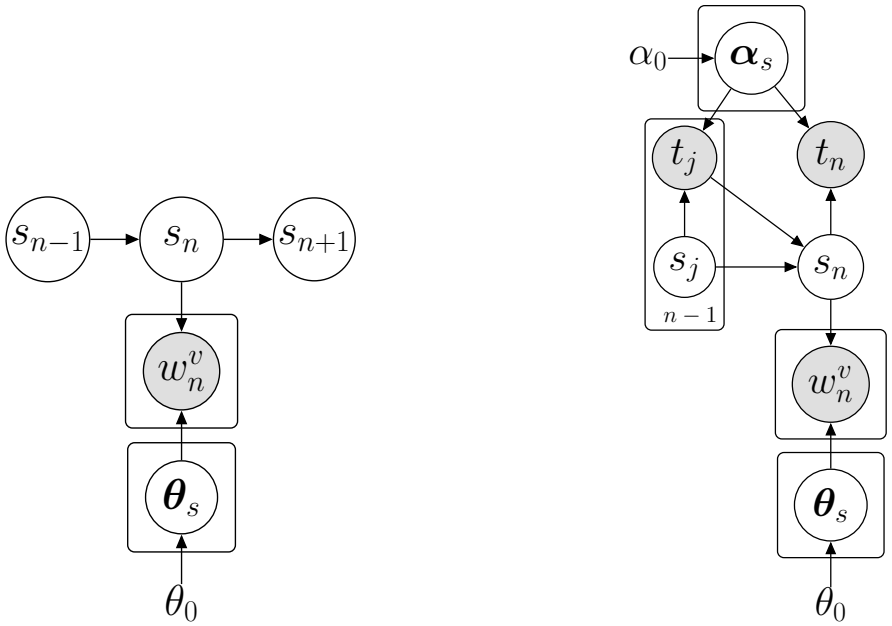
Therefore, the Dirichlet point process can capture the following four desirable properties:

1. Preferential attachment: Draw $\theta_n$ according to $\lambda_{\theta_k}(t_n)$. The larger the intensity for a Hawkes process, the more likely the next event is from that cluster.

2. Adaptive number of clusters: Draw $\theta_n$ according to $\lambda_0$. There is always some probability of generating new cluster with $\lambda_0$.

3. Self-excitation: This is captured by the intensity of the Hawkes process $\lambda_{\theta_k}(t) = \sum_{i=1}^{n-1} \gamma_{\theta_i}(t_n, t_i)\mathbb{I}[\theta_i = \theta_k]$.

4. Temporal decays: This is captured by the triggering kernel function $\gamma_\theta(t, t_i)$ which is typically decaying over time.




(a) Recurrent Chinese Restaurant Process (b) Dirichlet Point Process

Figure 8.2: Generative models of RCRP and DPP.

Finally, given the sequence of events (or samples) $\mathcal{T} = \{(t_i, \theta_i)\}_{i=1}^n$ from a Dirichlet Point process, the likelihood of the event arrival times can be evaluated as

$$\mathcal{L}(\mathcal{T}) = \exp\left(-\int_0^T \bar{\lambda}(\tau)d\tau\right) \prod_{(t_i,\theta_i)\in\mathcal{T}} \lambda_{\theta_i}(t_i), \tag{8.5}$$

Compared to the recurrent Chinese restaurant process (RCRP) appeared in [6], a distinctive feature of the Dirichlet Point process is that there is no need to discretize the time and divide events into episodes. Furthermore, the temporal dynamics is controlled by more general triggering kernel functions, and can be statistically learned from data.

## 8.4 Generating Text With DPP

As we have defined the Dirichlet point Process, we will use it as a prior for modeling continuous-time document streams. The goal is to discover clusters from the document stream based on both contents and temporal dynamics. Essentially, the set of $\{\theta_i\}$ sampled from the DPP will be used as the parameters for document content

model, and each cluster will have a distinctive value of $\theta_d \in \{\theta_i\}$. Furthermore, we will allow different clusters to have different temporal dynamics, with the corresponding triggering kernel drawn from a mixture of $K$ base kernels. We will first present the overall generative process of the model before going into details of these components in Figure 8.2(b).

1. Draw $t_1$ from Poisson($\lambda_0$), $\theta_1$ from Dir($\theta|\theta_0$), and $\alpha_{\theta_1}$ from Dir($\alpha|\alpha_0$).
2. For each word $v$ in document 1: $w_1^v \sim \text{Multi}(\theta_1)$
3. For $n > 1$:
   (a) Draw $t_n > t_{n-1}$ from Poisson$\big(\lambda_0 + \sum_{i=1}^{n-1} \gamma_{\theta_i}(t_n, t_i)\big)$,
   $$\text{where } \gamma_{\theta_i}(t_n, t_i) = \sum_{l=1}^{K} \alpha_{\theta_i}^l \cdot \kappa(\tau_l, t_n - t_i)$$
   (b) Draw $\theta_n$ from Dir($\theta|\theta_0$) with probability
   $$\frac{\lambda_0}{\lambda_0 + \sum_{i=1}^{n-1} \gamma_{\theta_i}(t_n, t_i)}, \tag{8.6}$$
   and draw $\alpha_{\theta_n}$ from Dir($\alpha|\alpha_0$)
   (c) Reuse previous $\theta_k$ for $\theta_n$ with probability
   $$\frac{\lambda_{\theta_k}(t_n)}{\lambda_0 + \sum_{i=1}^{n-1} \gamma_{\theta_i}(t_n, t_i)}, \tag{8.7}$$
   where $\lambda_{\theta_k}(t_n) = \sum_{i=1}^{n-1} \gamma_{\theta_i}(t_n, t_i)\delta(\theta_i = \theta_k)$.
   (d) For each word $v$ in document $n$:
   $$w_n^v \sim \text{Multi}(\theta_n)$$

**Content Model.** Many document content models can be used here. For simplicity of exposition, we have used a simple bag-of-word language model for each cluster in

the above generative process. In this case, the base distribution $G(\theta)$ in the DPP is now chosen as a Dirichlet distribution, $\mathrm{Dir}(\theta|\theta_0)$, with parameter $\theta_0$. Then, $w_n^v$, the $v$th word in the $n$th document is sampled according to a multinomial distribution

$$w_n^v \sim \mathrm{Multi}(\theta_{s_n}). \tag{8.8}$$

where $s_n$ is the cluster indicator variable for the $n$th document, and the parameter $\theta_{s_n}$ is a sample drawn from the DPP process.

**Triggering kernel.** We allow different clusters to have different temporal dynamics, by representing the triggering kernel function of the Hawkes Process as a non-negative combination of $K$ base kernel functions,*i.e.*,

$$\gamma_\theta(t_i, t_j) = \sum_{l=1}^{K} \alpha_\theta^l \cdot \kappa(\tau_l, t_i - t_j), \tag{8.9}$$

where $t_j < t_i, \sum_l \alpha_\theta^l = 1, \alpha_\theta^l > 0$, and $\tau_i$ is the typical reference time points, *e.g.*, 0.5, 1, 8, 12, 24 hours *etc.*

To simplify notations, define $\Delta_{ij} = t_i - t_j$, $\boldsymbol{\alpha}_\theta = (\alpha_\theta^1, \ldots, \alpha_\theta^K)^\top$ and $\boldsymbol{k}(\Delta_{ij}) = (\kappa(\tau_1, \Delta_{ij}), \ldots, \kappa(\tau_K, \Delta_{ij}))^\top$, so $\gamma_\theta(t_i, t_j) = \boldsymbol{\alpha}_\theta^\top \boldsymbol{k}(\Delta_{ij})$. Since each cluster has its own set of kernel parameters $\boldsymbol{\alpha}_\theta$, we are able to track their different evolving processes. Given $\mathcal{T} = \{(t_n, \theta_n)\}_{n=1}^N$, the intensity function of the cluster with parameter $\theta$ is represented as

$$\lambda_\theta(t) = \sum_{t_i < t} \boldsymbol{\alpha}_\theta^\top \boldsymbol{k}(t - t_i) \delta(\theta_i = \theta), \tag{8.10}$$

and the likelihood $\mathcal{L}(\mathcal{T})$ of observing the sequence $\mathcal{T}$ before time $T$ based on (8.5) is

$$\exp\left(-\sum_{\theta_i=\theta} \boldsymbol{\alpha}_\theta^\top \boldsymbol{g}_\theta - \Lambda_0\right) \prod_{\theta_i=\theta} \sum_{t_j<t_i, \theta_j=\theta} \boldsymbol{\alpha}_\theta^\top \boldsymbol{k}(\Delta_{ij}), \tag{8.11}$$

where $g_\theta^l = \sum_{t_i<T, \theta_i=\theta} \int_{t_i}^T \kappa(\tau_l, t - t_i) dt$ and $\Lambda_0 = \int_0^T \lambda_0 dt$. This can be done efficiently for many kernels, such as the Gaussian RBF kernel [44, 41], Rayleigh kernel [1],

*etc.* Here, we choose the Gaussian RBF kernel $\kappa(\tau_l, \Delta) = \exp(-(\Delta-\tau_l)^2)/2\sigma_l^2)/\sqrt{2\pi\sigma_l^2}$, so the integral $g_\theta^l$ has the analytic form:

$$\sum_{t_i<T,\theta_i=\theta} \frac{1}{2}\left(\operatorname{erfc}\left(-\frac{\tau_l}{\sqrt{2\sigma_l^2}}\right) - \operatorname{erfc}\left(\frac{T-t_i-\tau_l}{\sqrt{2\sigma_l^2}}\right)\right) \tag{8.12}$$

**Inexact event timing.** In practice, news articles are usually automatically collected and indexed by web crawlers. Sometimes, due to unexpected errors or the available minimum timing resolution, we can observe a few $m$ documents at the same time $t_n$. In this case, we assume that each of the $m$ documents actually arrived between $t_{n-1}$ and $t_n$. To model this rare situation, we can randomly pick $t_n$ and replace the exact timestamps within the interval $[t_n, t_{n+m-1}]$ by $t_{n+m-1}$ to take that into account.

## 8.5 Inference

Given a stream of documents $\{(d_i, t_i)\}_{i=1}^n$, at a high level, the inference algorithm alternates between two subroutines. The first subroutine samples the latent cluster membership (and perhaps the missing time) for the current document $d_n$ by Sequential Monte Carlo [39, 40]; and then, the second subroutine updates the learned triggering kernels of the respective cluster on the fly.

**Sampling the cluster label.** Let $s_{1:n}$ and $t_{1:n}$ be the latent cluster indicator variables and document time for all the documents $d_{1:n}$. For each $s_n$, we have $s_n \in \{0, 1, \ldots, D\}$, where $D$ is the total number of distinctive values $\{\theta_i\}$, and $s_n = 0$ refers to the background Poisson process Poisson($\lambda_0$). In the streaming context, it is shown by [3, 4] that it would be more suitable to efficiently draw a sample for the latent cluster labels $s_{1:n}$ shown in Figure 8.2(b) from $P(s_{1:n}|d_{1:n}, t_{1:n})$ by reusing the past samples from $P(s_{1:n-1}|d_{1:n-1}, t_{1:n})$, which motivates us to apply the Sequential Monte Carlo method [39, 40, 3, 4]. Briefly, a particle keeps track of an approximation of the posterior $P(s_{1:n-1}|d_{1:n-1}, t_{1:n-1})$, where $d_{1:n-1}$, $t_{1:n-1}$, $s_{1:n-1}$

represent all past documents, timestamps and cluster labels, and updates it to get an approximation for $P(s_{1:n}|d_{1:n}, t_{1:n})$. We maintain a set of particles at the same time, each of which represents a hypothesis about the latent random variables and has a weight to indicate how well its hypothesis can explain the data. The weight $w_n^f$ of each particle $f \in \{1, \ldots, F\}$ is defined as the ratio between the true posterior and a proposal distribution $w_n^f = \frac{P(s_{1:n}|d_{1:n},t_{1:n})}{\pi(s_{1:n}|d_{1:n},t_{1:n})}$. To minimize the variance of the resulting particle weight, we take $\pi(s_n|s_{1:n-1}, d_{1:n}, t_{1:n})$ to be the posterior distribution $P(s_n|s_{1:n-1}, d_{1:n}, t_{1:n})$ [40, 3]. Then, the unnormalized weight $w_n^f$ can be updated by

$$w_n^f \propto w_{n-1}^f \cdot P(d_n|s_n^f, d_{1:n-1}). \tag{8.13}$$

Because the posterior is decomposed as $P(s_n|d_n, t_n, \text{rest}) \sim P(d_n|s_n, \text{rest}) \cdot P(s_n|t_n, \text{rest})$, by the Dirichlet-Multinomial conjugate relation, the likelihood $P(d_n|s_n, \text{rest})$ is given by

$$\frac{\Gamma\left(C^{s_n \backslash d_n} + V\theta_0\right) \prod_v^V \Gamma\left(C_v^{s_n \backslash d_n} + C_v^{d_n} + \theta_0\right)}{\Gamma\left(C^{s_n \backslash d_n} + C^{d_n} + V\theta_0\right) \prod_k^K \Gamma\left(C_v^{s_n \backslash d_n} + \theta_0\right)}, \tag{8.14}$$

where $C^{s_n \backslash d_n}$ is the word count of cluster $s_n$ excluding the document $d_n$, $C^{d_n}$ is the word count of document $d_n$, $C_v^{s_n \backslash d_n}$ and $C_v^{d_n}$ refer to the count of the $v$th word, and $V$ is the vocabulary size. Finally, $P(s_n|t_n, \text{rest})$ is the prior given by the Dirichlet Point process (8.6) and (8.7) as

$$P(s_n = k|t_n, \text{rest}) = \begin{cases} \frac{\lambda_{\theta_k}(t_n)}{\lambda_0 + \sum_{i=1}^{n-1} \gamma_{\theta_i}(t_n, t_i)} & \text{if } k \text{ occupied} \\ \frac{\lambda_0}{\lambda_0 + \sum_{i=1}^{n-1} \gamma_{\theta_i}(t_n, t_i)} & \text{otherwise} \end{cases} \tag{8.15}$$

**Updating the triggering kernel.** Given $s_n$, we denote the respective triggering kernel $\boldsymbol{\alpha}_{\theta_{s_n}}$ by $\boldsymbol{\alpha}_{s_n}$ for brevity. By the Bayesian rule, the posterior is given by $P\left(\boldsymbol{\alpha}_{s_n}|\mathcal{T}_{s_n}\right) \sim P(\mathcal{T}_{s_n}|\boldsymbol{\alpha}_{s_n}) P(\boldsymbol{\alpha}_{s_n}|\alpha_0)$, where $\mathcal{T}_{s_n} = \{(t_i, s_i)|s_i = s_n\}$ is the set of events in cluster $s_n$. We can either update the estimation of $\boldsymbol{\alpha}_{s_n}$ by MAP for that the log-likelihood of (8.11) is concave in $\boldsymbol{\alpha}_{s_n}$. Alternatively, we can draw a set of samples

$\left\{\boldsymbol{\alpha}_{s_n}^i\right\}_{i=1}^N$ from the prior $P(\boldsymbol{\alpha}_{s_n}|\alpha_0)$ and calculate the weighted average:

$$\hat{\boldsymbol{\alpha}}_{s_n} = \sum_{i=1}^{N} w_i \cdot \boldsymbol{\alpha}_{s_n}^i, \tag{8.16}$$

where $w_i = P(\mathcal{T}_{s_n}|\boldsymbol{\alpha}_{s_n}^i)P(\boldsymbol{\alpha}_{s_n}^i|\alpha_0)/\sum_i P(\mathcal{T}_{s_n}|\boldsymbol{\alpha}_{s_n}^i)P(\boldsymbol{\alpha}_{s_n}^i|\alpha_0)$. For simplicity, we choose the latter method in our implementation.

**Sampling the missing time.** In the rare case when $m$ documents arrive with the same timestamp $t_n$, the precise document time is missing during the interval $[t_{n-1}, t_n]$. As a result, we need joint samples for $\{(s_n^i, t_n^i)\}_{i=1}^m$ where $s_n^i$ and $t_n^i$ are the cluster membership and the precise arriving time for the $i$th document $d_n^i$. However, since $m$ is expected to be small in practice, we can use Gibbs sampling to draw samples from the distribution $P(t_n^{1:m}, s_n^{1:m}|t_{1:n-1}, s_{1:n-1}, \text{rest})$ where $s_n^{1:m}$ and $t_n^{1:m}$ are the cluster labels and document time for the $m$ documents in the current $n$th interval. The initial values for $t_n^{1:m}$ can be assigned uniformly from the interval $[t_{n-1}, t_n]$. After fixing the sampled $s_n^{1:m}$ and the other $\left\{t_n^k\right\}_{k\neq i}$, we are going to draw a new sample $t_n^{i\prime}$ from $P(t_n^i|s_n^i, \text{rest})$. Let $\mathcal{T}_{s_n^i}\backslash t_n^i$ be the set of all the document time excluding $t_n^i$ in cluster $s_n^i$. The posterior of $t_n^i$ is proportional to the joint likelihood $P(t_n^i|s_n^i, \mathcal{T}_{s_n^i}\backslash t_n^i, \text{rest}) \propto P(t_n^i, \mathcal{T}_{s_n^i}\backslash t_n^i|s_n^i, \text{rest})$. Therefore, we can apply Metropolis algorithm in one dimension to draw the next sample $t_n^{i\prime}$. Specifically, let's first uniformly draw $t_n^{i\prime}$ from $[t_{n-1}, t_n]$ and calculate the following ratio between the two joint likelihoods $r = \frac{P(t_n^{i\prime}, \mathcal{T}_{s_n^i}\backslash t_n^i|s_n^i, \text{rest})}{P(t_n^i, \mathcal{T}_{s_n^i}\backslash t_n^i|s_n^i, \text{rest})}$. We then accept $t_n^{i\prime}$ if $r > 1$; otherwise, we accept it with the probability $r$. With the new sample $t_n^{i\prime}$, we can update the kernel parameter by (8.16). Finally, we need to update the particle weight by considering the likelihood of generating such $m$ documents as

$$w_n^f \propto w_{n-1}^f \times \prod_{i=1}^{m} P(d_n^i|s_n^{f,i}, d_{1:n-1}, d_n^{1:m\backslash i}, \text{rest}) \times \prod_{s\in\{s_n^i\}_{i=1}^m} P(m_s|\mathcal{T}_s, \text{rest}), \tag{8.17}$$

where $d_n^{1:m\backslash i}$ is the set of $m$ documents excluding $d_n^i$, $m_s$ is the number of documents with cluster membership $s$ among the $m$ documents, and $\mathcal{T}_s$ is the set of document

**Algorithm 8.1:** The SMC Framework

Initialize $w_1^f$ to $\frac{1}{F}$ for all $f \in \{1 \dots F\}$;
**for** *each event time* $t_n, n = 1, 2, \dots$ **do**
  **for** $f \in \{1, \dots, F\}$ **do**
    **if** *one document* $d_n$ *at the time* $t_n$ **then**
      sample $s_n$ from (8.15) and add $t_n$ to $s_n$;
      update the triggering kernel by (8.16);
      update the particle weight by (8.13);
    **else if** $m > 1$ *documents* $d_n^{1:m}$ *with the same* $t_n$ **then**
      sample $\{s_n^{1:m}\}, \{t_n^{1:m}\}$ by Algorithm 8.2;
      update the particle weight by (8.17);
    **end**
  **end**
  Normalize particle weight;
  **if** $\|w_n\|_2^{-2} <$ *threshold* **then**
    resample particles;
**end**

time in cluster $s$. Conditioned on the history up to $t_n$, the Hawkes process is an inhomogeneous Poisson process, and thus we know that $P(m_s|\mathcal{T}_s, \text{rest})$ is simply a Poisson distribution with mean $\Lambda_s = \int_{t_{n-1}}^{t_n} \lambda_s(t)dt$. For Gaussian kernels, we can use (8.12) to obtain the analytic form of $\Lambda_s$. The overall pseudocode for Sequential Monte Carlo is formally presented in Algorithm 8.1, and the Gibbs sampling framework is given by Algorithm 8.2.

**Efficient Implementation.** In order to scale with large datasets, the online inference algorithm should be able to process each individual document in an expected constant time. Particularly, the expected time cost of sampling the cluster label and updating the triggering kernel should not grow with the amount of documents we have seen so far. The most fundamental operation in Algorithm 8.1 and 8.2 is to evaluate the joint likelihood (8.11) of all the past document time to update the triggering kernel in every cluster. A straightforward implementation requires repeated computation of a sum of Gaussian kernels over the whole history, which tends to be quadratic to the number of past documents.

**Algorithm 8.2:** Gibbs sampling for cluster label and time

**for** *iter* $= 1$ *to* $MaxIterG$ **do**
  **if** *update cluster label* $s_n^i$ **then**
    remove $t_n^i$ from cluster $s_n^i$ and update the triggering kernel by (8.16);
    draw a new sample $s_n^{i\prime}$ from (8.15);
    add $t_n^i$ into cluster $s_n^{i\prime}$ and update the triggering kernel by (8.16);
  **else if** *update document time* $t_n^i$ **then**
    **for** *iter* $= 1$ *to* $MaxIterM$ **do**
      draw a new sample $t_n^{i\prime} \sim \text{Unif}(t_{n-1}, t_n)$;
      **if** $r = \frac{P(t_n^{i\prime}, \mathcal{T}_{s_n^i} \setminus t_n^i | s_n^i, rest)}{P(t_n^i, \mathcal{T}_{s_n^i} \setminus t_n^i | s_n^i, rest)} > 1$ **then** $t_n^i \leftarrow t_n^{i\prime}$;
      **else** $t_n^i \leftarrow t_n^{i\prime}$ with probability $r$;
    **end**
    update the triggering kernel of $s_n^i$ by (8.16);
  **end**
**end**

Based on the fast decaying property that the Gaussian kernel decreases exponentially as the distance deviating from its center increases quadratically, we can alleviate the problem by ignoring those past time far away from the kernel center. Specifically, given an error tolerance $\epsilon$, we only need to look back until we reach the time

$$t_u = t_n - \left( \tau_m + \sqrt{-2\sigma_m \log\left(0.5\epsilon\sqrt{(2\pi\sigma_m^2)}\right)} \right), \tag{8.18}$$

where $\tau_m = \max_l \tau_l$, $\sigma_m = \max_l \sigma_l$ and $t_n$ is the current document time to guarantee that the error of the Gaussian summation with respect to each reference point $\tau_l$ is at most $\epsilon$. Because the number of documents within $[t_u, t_n]$, referred to as the *active interval*, is expected to be constant as we run the algorithm for a while when the Hawkes Process becomes stationary, the average running time will keep stable in the long run. In addition, from the log of (8.11), for the newly added time $t_n$, we only need to add the new intensity value $\lambda_{s_n}(t_n)$, set the observation window $T = t_n$, and update the integral of the intensity function (8.12). Therefore, we can precompute and store the likelihood value for each sample $\boldsymbol{\alpha}_{s_n}^k$ and incrementally update it in each cluster. Similarly, in the Metropolis loop of Algorithm 8.2, we need to update the

triggering kernel whenever a document time $t_j$ is updated or deleted from a cluster. In this case, since the observation window $T$ is fixed, we only need to recompute the affected intensity value $\lambda_{s_n}(t_j)$ and the affected individual summation terms in (8.12) for those time $t_i < t_n, t_i \in \mathcal{T}_{s_n}$.

In a nutshell, because we depend on the past documents only within the *active interval*, the above partial updating only performs the necessary calculations, and the overall memory usage and the expected time cost per document tend to be constant with respect to the number of incoming documents and the existing number of clusters, which is empirically verified in Figure 8.6 of the following experiments.

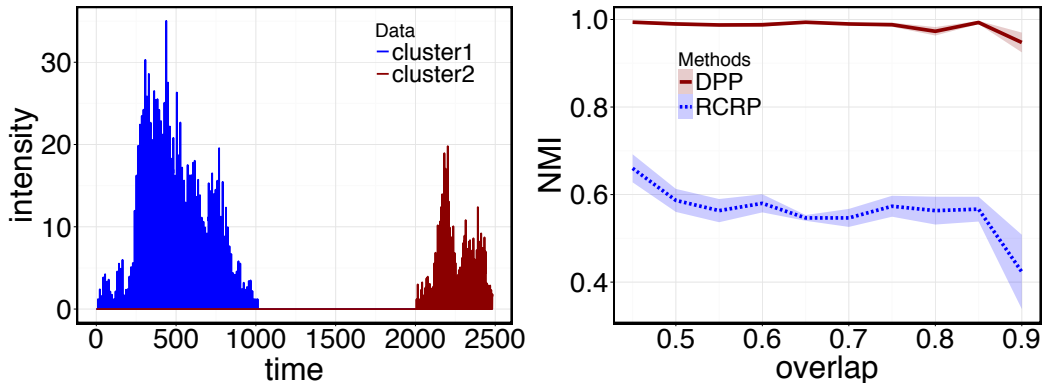
## 8.6 Experiments

On massive synthetic and real-world datasets, in this section, we demonstrate that DPP not only can provide clusters of relevant news articles but also is able to uncover meaningful latent temporal dynamics inherent inside those clusters.

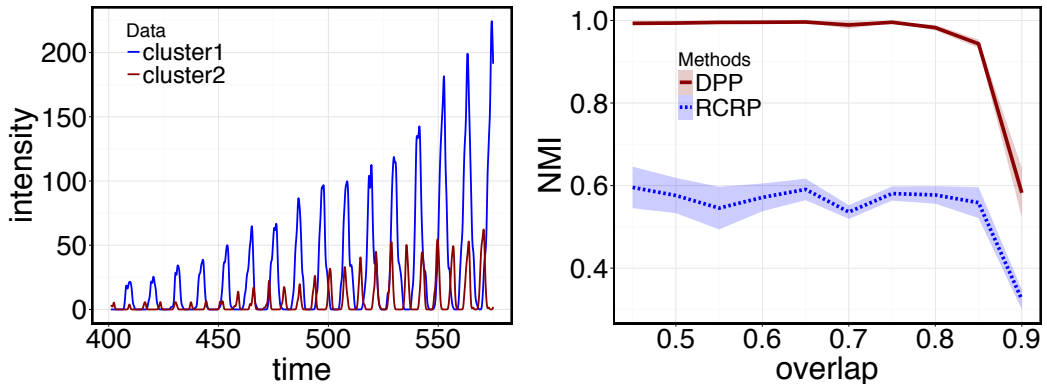
### 8.6.1 Synthetic Data

On synthetic data, we investigate the effectiveness of the temporal dynamics for improving clustering, the learning performance of the inference algorithm and the efficacy of the sampling method for missing time.

**Do temporal dynamics help?** Because DPP exploits both temporal dynamics and textual contents, we expect that documents with similar topics and temporal patterns should be related to each other. On the other hand, for those documents with similar topics but different temporal behaviors, our model should still be able to disambiguate them to certain extent. We simulate two clusters on a vocabulary set of 10,000 words. The word distribution of one cluster mainly concentrates on the first 8,000 words, and we shift the word distribution of the other one to have a varying vocabulary overlap from 45 to 90 percent. The triggering kernels of the clusters have




(a) Temporally well-separated clusters.




(b) Temporally interleaved clusters.

Figure 8.3: Effectiveness of Temporal Dynamics. Panel (a) and (b) show different cases where the clusters are temporally well-separated and interleaved, respectively. In each case, the left plot shows the intensity function of each cluster, and the right plot compares the performance by Normalized Mutual Information.

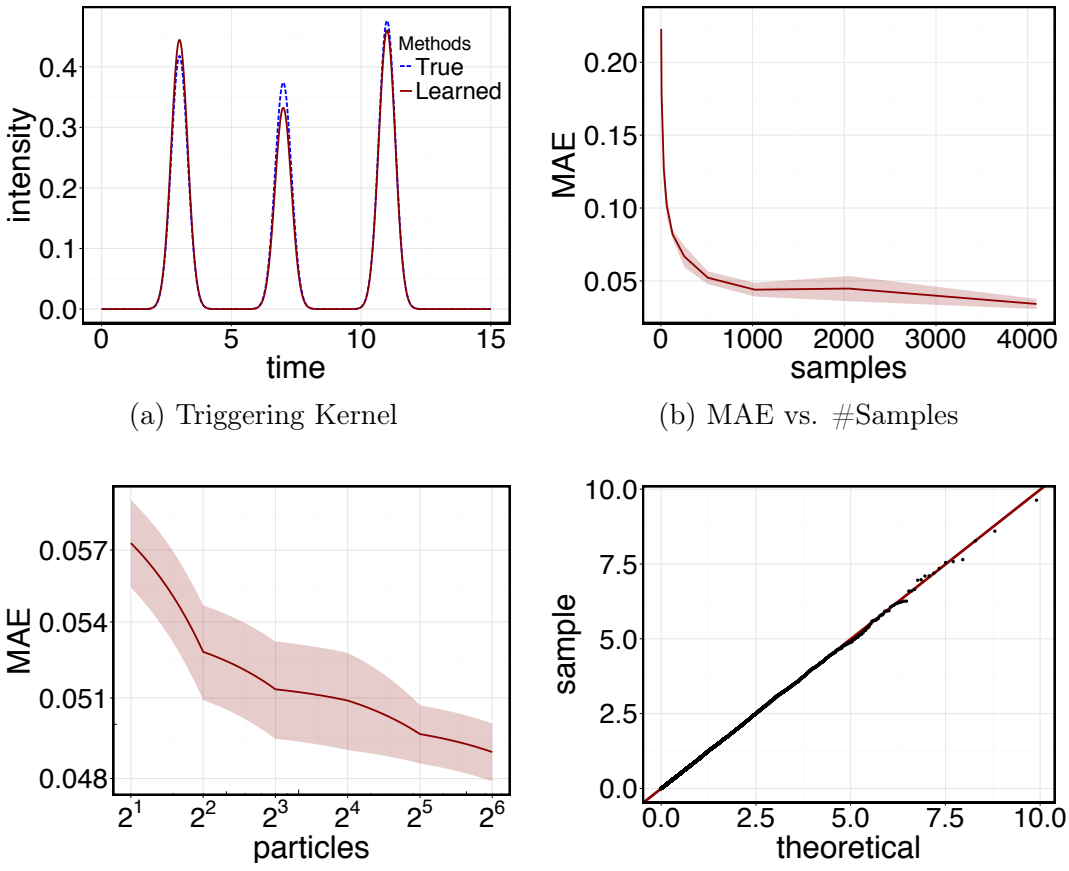
two basic RBF kernels at 7 and 11 on the time line with bandwidth 0.5. We set $\alpha_0 = 1$ of the language model for both methods, and set $\lambda_0 = 0.01$ for DPP. We use the Normalized Mutual Information (NMI) to compare the uncovered clusters with the ground-truth clusters. The range of NMI is from 0 to 1, so larger values indicate better performance. All experiments are repeated for 10 times.

In Figure 8.3(a), we first consider an easy case where the clusters are well-separated in time, which corresponds to the usual case that each cluster corresponds to a single short lifetime event. Because the clusters come and go sequentially in time, it helps to differentiate the clusters as their topics become more similar. This effect is verified

in the right panel of Figure 8.3(a) where the NMI value is still close to one even when the topic vocabularies have 90% overlap. Besides, in Figure 8.3(b), we consider a more challenging situation where the clusters evolve side-by-side in time. Because the clusters have different triggering kernels, we can still expect the Dirichlet-Hawkes model to perform well. In the right panel of Figure 8.3(b), the performance of DPP only starts to decrease when the overlapping grows to 85-percent. Overall, because RCRP does not explicitly learn the temporal dynamics of each cluster, it cannot tell the temporal difference. In contrast, as DPP clusters the incoming documents, it also automatically updates its inference about the temporal dynamics of each cluster, and Figure 8.3 demonstrates that this temporal information could be useful to have better clustering performance.

**Can we learn temporal dynamics effectively?** Without loss of generality, each cluster has RBF kernels located at 3, 7, and 11 with bandwidth 0.5. We let true coefficients of the triggering kernels for each cluster be uniformly generated from the simplex and simulated 1,000,000 documents. We randomly produce the missing time to allow at most three documents to arrive at the same time. The maximum Gibbs and Metropolis iteration is set to 100 and 50 with 8 particles in total. Figure 8.4(a) shows the learned triggering kernel for one randomly chosen cluster against the ground-truth. Because the size of the simulated clusters is often skew, we only compare the estimated triggering kernels with the ground-truth for the top-100 largest clusters. Moreover, Figure 8.4(b) presents the estimation error with respect to the number of samples drawn from the Dirichlet prior. As more samples are used, the estimation performance improves, and Figure 8.4(c) shows that only a few particles are enough to have good estimations.

**How well can we sample the missing time?** Finally, we check whether the sampled missing time from Algorithm 8.2 are valid samples from a Hawkes Process.




(a) Triggering Kernel

(b) MAE vs. #Samples

(c) MAE vs. #Particles

(d) Quantile Plot

Figure 8.4: (a) Learned triggering kernels of one cluster from 1,000,000 synthetic documents; (b) Mean absolute error decreases as more samples are used for learning the triggering kernels; (c) A few particles are sufficient to have good estimation; (d) Quantile plot of the intensity integrals from the sampled document time.

Fixing an observation window $T = 100$, we first simulate a sequence of events $\mathcal{H}_T$ with the true triggering kernels. Given $\mathcal{H}_T$, the form of the intensity function $\lambda(t|\mathcal{H}_T)$ is fixed. Next, we equally divide the interval $[0, T]$ into five partitions $\{\mathcal{T}_i\}_{i=1}^5$, in each of which we incrementally draw $\Lambda_{\mathcal{T}_i} = \int_{\mathcal{T}_i} \lambda(t|\mathcal{H}_T)dt$ samples by Algorithm 8.2 using the true kernels. Then, we collect the samples from all partitions to see whether this new sequence is a valid sample from the Hawkes Process with the intensity $\lambda(t|\mathcal{H}_T)$. By the Time Changing Theorem [35], the intensity integrals $\int_{t_{i-1}}^{t_i} \lambda(\tau)d\tau$ from the sampled sequence should conform to the unit-rate exponential distribution. Figure 8.4(d) presents the quantiles of the intensity integrals against the quantiles of

the unit-rate exponential distribution. It clearly shows that the points approximately lie on the line indicating that the two distributions are very similar to each other and thus verifies that Algorithm 8.2 can effectively generate samples for the missing time from the Hawkes Process of each cluster.

### 8.6.2 Real Data

We further examine our model on a set of 1,000,000 mainstream news articles extracted from the Spinn3r[2] dataset from 01/01 to 02/15 in 2011.

**Setup.** We apply the Named Entity Recognizer from Stanford NER system [51] and remove common stop-words and tokens which are neither verbs, nouns, nor adjectives. The vocabulary of both words and named entities is pruned to a total of 100,000 terms. We formulate the triggering kernel of each cluster by placing a RBF kernel at each typical time point: 0.5, 1, 8, 12, 24, 48, 72, 96, 120, 144 and 168 hours with the respective bandwidth being set to 1, 1, 8, 12, 12, 24, 24, 24, 24, 24, and 24 hours, in order to capture both the short-term and long-term excitation patterns. To enforce the sparse structure over the triggering kernels, we draw 4,096 samples from the Dirichlet prior with the concentration parameter $\alpha_0 = 0.1$ for each cluster. The intensity rate for the background Poisson process is set to $\lambda_0 = 0.1$, and the Dirichlet prior of the language model is set to $\phi_0 = 0.01$. In fact, the results are robust across a wide range of settings from 0.01 to 0.1 for both $\theta_0$ and $\lambda_0$, which can be further tuned by following the techniques in [3]. We report the results by using eight particles.

**Content Analysis.** Figure 8.5 shows four discovered example stories, including the 'Tucson shooting' event[3], the movie 'Dark Knight Rises[4]', Space Shuttle Endeavor's

[2] http://www.icwsm.org/data/
[3] http://en.wikipedia.org/wiki/2011_Tucson_shooting
[4] http://www.theguardian.com/film/filmblog/2011/jan/13/batman-dark-knight-rises

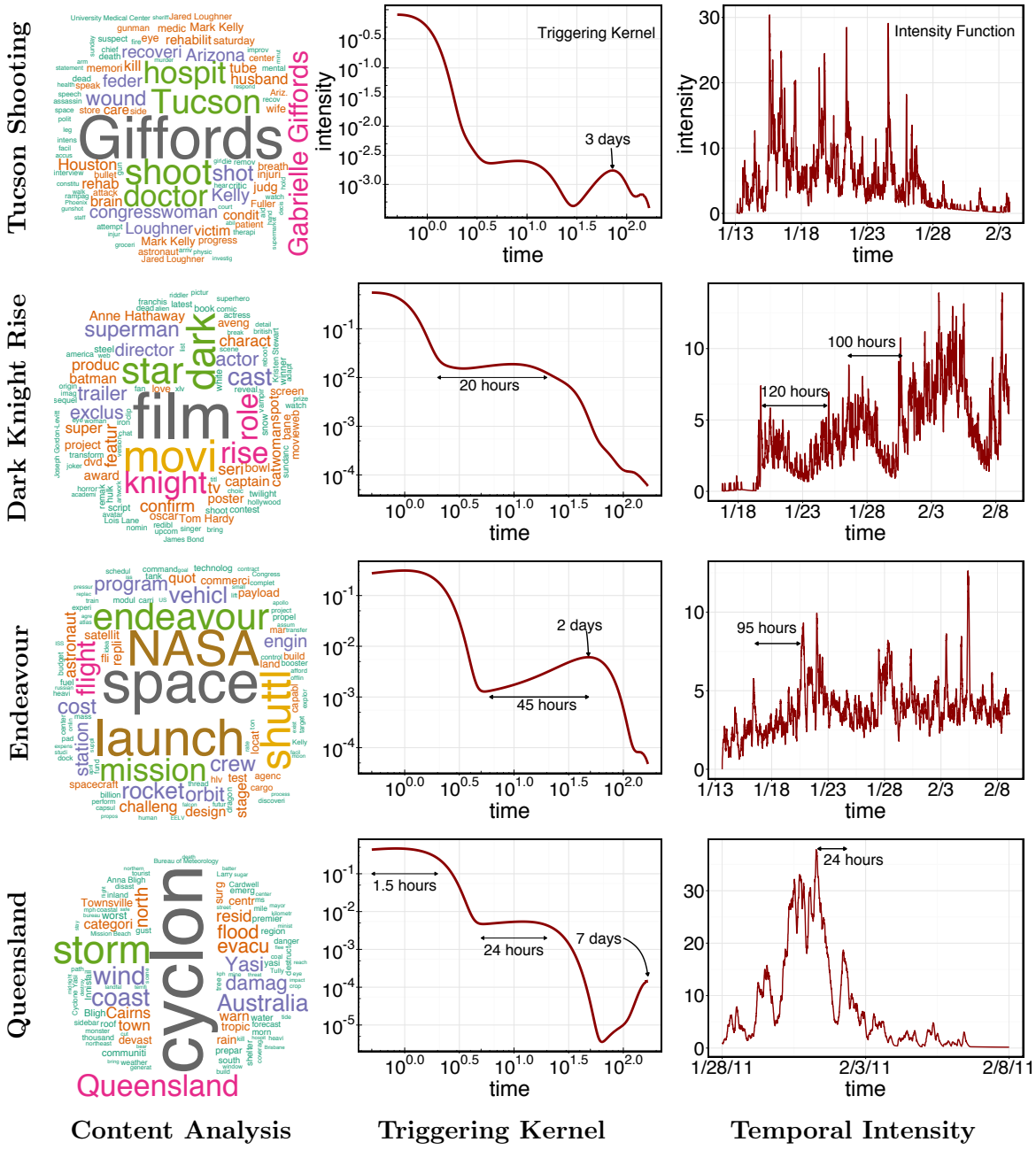



Figure 8.5: Four example stories extracted by our model, including the 'Tucson Shooting' event, the movie of 'Dark Knight Rises', Space Shuttle's final mission and Queensland flooding disaster. For each story, we list the top 100 most frequent words on the left column. The middle column shows the learned triggering kernel in the log-log scale, and the right column presents the respective intensity functions along time.

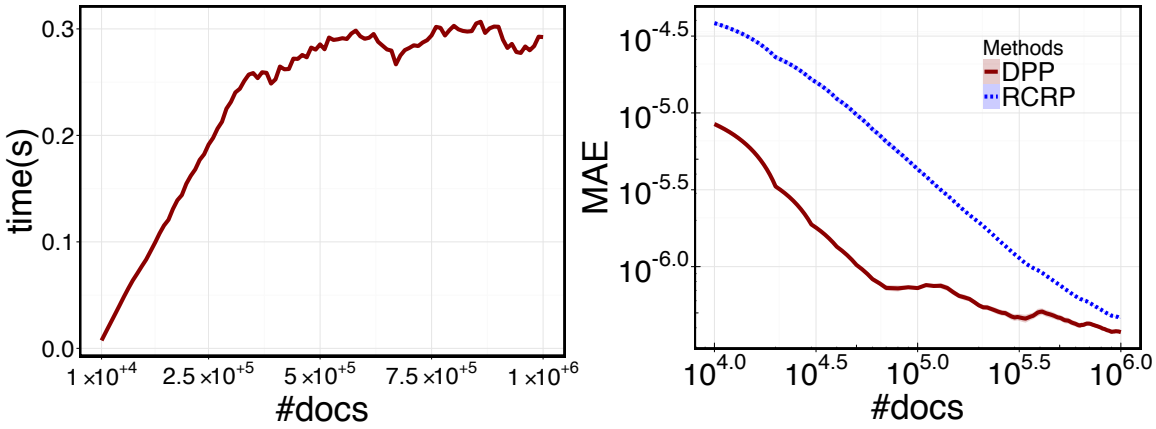
last mission[5], and 'Queensland Flooding' disaster[6]. The leftmost column lists the

[5]http://en.wikipedia.org/wiki/Space_Shuttle_Endeavour
[6]http://en.wikipedia.org/wiki/Cyclone_Yasi

top-100 frequent words in each story, showing that DPP can deduce the clusters with meaningful topics.

**Triggering Kernels.** The middle column of Figure 8.5 gives the learned triggering kernel of each story, which quantifies the influence over future events from the occurrence of the current event. For the 'Tucson Shooting' story, its triggering kernel reaches the peak within half an hour since its birth, decays quickly until the 30th hour, and then has a weak tailing influence around the 72nd hour, showing that it has a strong short-term effect, that is, most related articles and posts arrive closely in time. In contrast, the triggering kernel of the story 'Dark Knight Rises' keeps stable for around 20 hours before it decays below $10^{-4}$ by the end of a week. The continuous activities of this period indicate that the current event tends to have influence over the events 20 hours later.

**Temporal Dynamics.** The rightmost column of Figure 8.5 plots the respective intensity functions which indicate the popularity of the stories along time. We can observe that most reports of 'Tucson Shooting' concentrate within the following two weeks starting from 01/13/2011 and fade out quickly by the end of the month. In contrast, we can verify the longer temporal effect of the 'Dark Knight Rises' movie in the second row of Figure 8.5 where the temporal gaps between two large spikes are about several multiples of the 20-hour period. Because this story is more about entertainment, including the articles about Anne Hathaway's playing of the Cat-woman in the film as well as other related movie stars, it maintains a certain degree of hotness by attracting people's attention as more production details of the movie are revealed. For the NASA Endeavour event we can see in the intensity function of the third row in Figure 8.5 the elapsed time between two observed large spikes is around a multiple of 45-hour, which is also consistent with its corresponding triggering kernel. Finally, for the event of 'Queensland Flooding', the 'Cyclone Yasi' intensified to a Category




(a) Scalability (b) Time prediction

Figure 8.6: Scalability and time prediction in real world news stream.

3 cyclone on 01/31/2011, to a Category 4 on 02/01/2011, and to a Category 5 on 02/02/2011[7]. These critical events again coincide with the observed spikes in the intensity function of the story in the bottom row of Figure 8.5. Because the intensity functions depend on both the triggering kernels and the arriving rate of news articles, news reports of emergent incidents and disasters tend to be concentrated in time to form strong short-term clusters with higher magnitude of intensity values. In Figure 8.5, the intensity functions of both 'Tucson Shooting' and 'Queensland Flooding' have value greater than 20. In contrast, other types of stories in entertainment and scientific explorations might have continuous longer-term activities as more and more related details get revealed. Overall, the ability of uncovering topic-specific clusters with learned latent temporal dynamics of our model provides a better and intuitive way to track the trend of each evolving story in time.

**Scalability.** Figure 8.6(a) shows the scalability of our learning algorithm. Since the number of clusters grows logarithmically as the number of data points increases for CRP, we expect the average time cost of processing each document is keeping roughly constant after running for a long time period. This is verified in Figure 8.6(a) where

[7]http://en.wikipedia.org/wiki/Cyclone_Yasi

after the build-up period, the average processing time per 10,000-document keeps stable.

**Prediction.** Finally, we evaluate how well the learned temporal model of each cluster can be used for predicting the arrival of the next event. Starting from the 5,000th document, we predict the possible arriving time of the next document for the clusters with size larger than 100. Since RCRP does not learn the temporal dynamics, we use the average inter-event gap between two successive documents as the predicted time interval between the most recent document and the next one in the future. For DPP, we simulate the next event time based on the learned triggering kernels and the timestamps of the documents observed so far. We treat the average of five simulated time as our final prediction and report the cumulative mean DPP absolute prediction error in Figure 8.6(b) in the log-log scale. As more documents are observed, the prediction errors of both methods decrease. However, the prediction performance of DPP is even better from the very beginning when the number of documents is still relatively small, showing that the Hawkes model indeed can help to capture the underlying temporal dynamics of the evolution of each cluster.

## 8.7 Summary

In addition to RCRP, several other well-known processes can also be incorporated into the framework of DPP. For instance, we may generalize the Pitman-Yor Process [156] to incorporate the temporal dynamics. This simply brings back the constant rate for each Hawkes Process. A small technical issue arises from the fact that if we were to decay the counts $m_{k,t}$ as in the RCRP, we would obtain negative counts from $m_{k,t} - a$, where $a$ is the parameter of the Pitman-Yor Process to increase the skewness of the cluster size distribution. However, this can be addressed, e.g., by clipping the terms by 0 via $\max(0, m_{k,t})$. In this form we obtain a model that further

encourages the generation of new topics relative to the RCRP. Moreover, the Distance-Dependent Chinese Restaurant Process (DD-CRP) of [22] attempts to address spatial interactions between events. This generalizes the CRP and, with a suitable choice of distance function, can be shown to contain the RCRP as a special case. The same notion can be used to infer spatial / logical interactions between Hawkes Processes to obtain spatiotemporal effects. That is, we simply use spatial excitation profiles to model the rate of each event.

To summarize, we establish a previously unexplored connection between Bayesian Nonparametrics and Temporal Point Processes, which allows the number of clusters to grow in order to accommodate the increasing complexity of online streaming contents, while at the same time learns the ever changing latent dynamics governing the respective continuous arrival patterns inherently. We point out that our combination of Dirichlet processes and Temporal Point Processes has implications beyond clustering streaming documents. We will show that our construction can be generalized to other Nonparametric Bayesian models, such as the Pitman-Yor processes [156] and the Indian Buffet processes [63]. We propose an efficient online inference algorithm which can scale up to millions of news articles with near constant processing time per document and moderate memory consumption. Experiments on both synthetic and real world news data demonstrate that by explicitly modeling the textual content and the latent temporal dynamics of each cluster, it provides an elegant way to uncover topically related documents and track their evolutions in time simultaneously.

# CHAPTER IX

# MULTIVARIATE POINT PROCESS PACKAGE

In this chapter, we describe a C++ software library, `PtPack`, for learning and making inference from high-dimensional point processes. This package provides a basic implementation of our proposed framework for fitting and assessing general large-scale multivariate point processes with different structure constraints, including: sparse structure, group sparse structure, low-rank structure, *etc.* In addition to the standard simulation task, it also supports the various inference algorithms proposed in the thesis, such as scalable inference estimation and maximization, time-sensitive recommendation, shaping user activities, *etc.*

## 9.1 Introduction

Many open source packages written in R are available on CRAN for analyzing event data. For instance, the `spatstat` [13, 14] and the `splancs` [140] package are developed for analyzing two-dimensional spatial point processes. The `PtProcess` [67, 68] package includes a wide class of marked and non-marked lower-dimensional temporal point processes with main seismological applications. The `PtProc` package builds upon the earlier version of `PtProcess` and extends it to the multivariate case.

The choice of each package depends on the application of interest. For example, `spatstat` is very good at modeling two-dimensional spatial point processes, but the models provided do not capture the long-term dependency over the history, and they are mainly depend on the specific Papangelou conditional intensity function [68]. Although `PtProc` emphasizes its capability to handle the general multi-dimensional cases, it fails to capture the various structural constraints over the dimensions and

support few inference tasks. Furthermore, since R is an interactive computing environment, these packages often cannot scale up to large datasets because of the efficiency limits of R in computation.

As a consequence, the `PtPack` library herein provides efficient implementation of fitting and assessing general high-dimensional temporal point processes under the various structural constraints, including: sparsity, group sparsity, low-rank, *etc.*, as proposed in the framework of the thesis. More remarkably, it also supports many fundamental inference tasks, like the scalable influence estimation, time-sensitive recommendation and the activity shaping, based on the learned models for making time-critical decisions. The implemented general routines for specifying customized processes and the open-source nature of `PtPack` makes it easy for researchers in the machine learning and data mining community to contribute additional functionalities, and help people to conduct future research around various temporal dynamics arising from systems of networked interactive entities.

## 9.2 Program Structure and Implementation

`PtPack` is built upon the Eigen library [1] and is compatible with Linux, Mac OS and Windows. The package consists of the following seven major components, as shown in Figure 9.1.

**Data** encapsulates the functions of loading and storing events. Essentially, each sequence has a unique ID and comprises a vector of events. It supports very basic operations, such as adding a new event, getting the observation window $T$ associated with this sequence, *etc.* Each event in turn has a unique event ID and includes the time, the marker, the dimension ID this event occurs to, the sequence ID this event is associated with, *etc.*

[1] http://eigen.tuxfamily.org

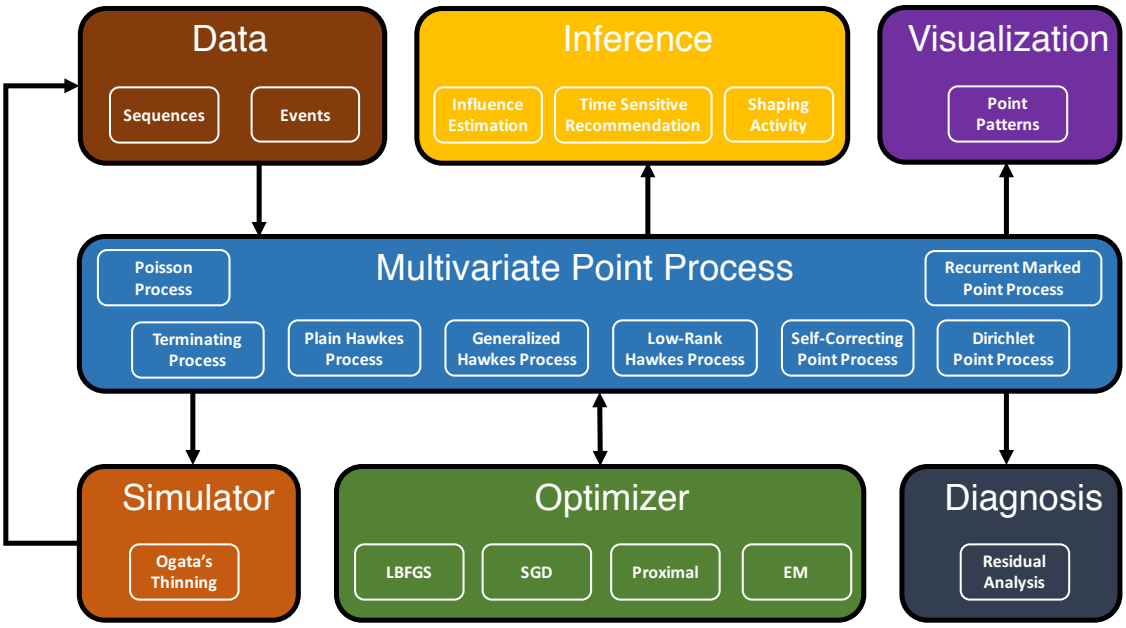



Figure 9.1: Architect of `PtPack` Library.

**Multivariate Point Process** defines the interface of a general high-dimensional temporal point process. In addition to the standard Poisson process [88], Hawkes process [72], Self-Correcting process [79], Terminating process, it supports the customized temporal point process and triggering kernels as long as the user implements the required interface of the conditional intensity function, the upper bound of the conditional intensity and the integral of the conditional intensity. This Multivariate Point Process component is the core part of `PtPack`. It provides the conditional intensity function and its upper bound for the Simulator component to generate synthetic events, and for the Visualization component to plot simple point patterns. It feeds the particular model parameters for the Inference component to conduct the respective inference tasks. It initializes a given process from the sequences provided by the Data component and then calculates the log-likelihood and the respective gradients for the Optimizer component to fit the model parameters. Finally, it also computes the integral of the conditional intensity function for the Diagnosis component to perform the residual analysis.

**Simulator** implements Ogata's Thinning Algorithm [124] to support three basic functionalities: I. simulate a sequence of events before a given time $T$; II. simulate a sequence consisting of a predefined number of events; and III. simulate the next event given an existing sequence of past events.

**Optimizer** provides four basic optimization procedures: a projected LBFGS algorithm [143], a general stochastic gradient descend method, the proximal gradient methods [41, 45] and the EM based algorithm [175]. It currently supports L1, L2, Group-Lasso [41] and nuclear norm [45] type of regularizations. Each individual optimization algorithm only requires the objective function value and the gradient of a vector of parameters returned from the Multivariate Point Process component. Based on the existing high performance computing techniques, such as OpenMP[2] and MPI[3], we are able to speed up the learning algorithms to large-scale datasets.

**Diagnosis** builds an approximate homogeneous Poisson process based on the integral of the conditional intensity function provided from the Multivariate Point Process component. It calculates an estimated intensity function for such constructed homogeneous Poisson process. By the Time Changing Theorem [35], if the estimated intensity value gets close to one, it means the assumed point process can better fit the given event data. Theoretically, any deviation of the residual process from a homogeneous Poisson process can be regarded as a deviation of the assumed model from the true process that generate the observed event samples.

**Visualization** gives basic general purpose figures for inspecting the characteristics of a given point process by plotting its conditional intensity function, the learned triggering kernels, the quantiles from the respective residual analysis, *etc.*

[2]http://openmp.org/wp/
[3]https://www.open-mpi.org

## 9.3 Basic Usage Examples

In this section, we give simple specific examples around the multivariate Hawkes process to demonstrate some basic functionalities of `PtPack`.

### 9.3.1 Simulation

We first simulate 10 sequences from a simple 2-d Hawkes process, each of which comprises 2,000 events. The code for the simulation is as follows:

```
// set the number of dimensions (dim) to be 2;
unsigned dim = 2, num_params = dim * (dim + 1);
// Initialize model parameters as a single vector;
// First two elements of params are the base intensity; The rest four elements are
    the excitation matrix unrolled by the column-wise order
Eigen::VectorXd params(num_params);
params << 0.1, 0.2, 0.5, 0.5, 0.5, 0.5;
// Initialize the decaying rates for the exponential triggering kernels;
Eigen::MatrixXd beta(dim,dim);
beta << 1, 1, 1, 1;
// Initialize a Hawkes process;
PlainHawkes hawkes(num_params, dim, beta);
hawkes.SetParameters(params);
// Store the simulated sequences;
std::vector<Sequence> sequences;
// Initialize the simulator;
OgataThinning ot(dim);
// Simulate 10 events for each sequence
unsigned n = 2000;
// Simulate 2 sequences
unsigned num_sequences = 10;
ot.Simulate(hawkes, n, num_sequences, sequences);
```

Code Snippet 9.1: Simple Hawkes Simulation

In the above code, we initialize an object of OgataThinning class which is used for simulating a general point process. Alternatively, for the specific plain Hawkes process, we can exploit the property of exponential triggering kernels [157] to make

```
1. Simulating 10 sequences with 1000 events each
2. Fitting Parameters

 Iteration        FunEvals      Step Length     Function Val      Opt Cond
         1               3      1.83276e-05       -1009.66         552.415
         2               4                1       -1010.88         538.709
         3               5                1       -1037.22         85.4665
         4               6                1        -1038.2         35.5692
         5               7                1        -1038.5         29.9313
         6               8                1       -1039.16         38.0669
         7               9                1       -1039.87         39.9242
         8              10                1          -1040         49.9864
         9              11                1       -1040.31         7.41528
        10              12                1       -1040.32         2.26413
        11              13                1       -1040.32        0.626209
        12              14                1       -1040.32        0.448012
        13              15                1       -1040.32        0.261424
        14              16                1       -1040.32        0.209407
        15              17                1       -1040.32       0.0649235
        16              18                1       -1040.32       0.0364096
        17              19                1       -1040.32       0.0322688
        18              20                1       -1040.32       0.0350413
        19              21                1       -1040.32       0.0417837
        20              22                1       -1040.32       0.0465589
        21              23                1       -1040.32       0.0441615
        22              24                1       -1040.32       0.0279519
        23              25                1       -1040.32       0.0109628
        24              26                1       -1040.32      0.00217949
        25              27                1       -1040.32     0.000138222
Directional Derivative below optTol

Estimated Parameters :
0.0872948  0.188331  0.497435  0.495623   0.46376  0.524999
True Parameters :
0.1 0.2 0.5 0.5 0.5 0.5
```

Figure 9.2: Demo results for fitting a 2-d Hawkes process with `PtPack`.

the simulation procedure more efficient based on dynamic programming, which scales linearly in the number of events. Therefore, we can also use the following code to simulate a Hawkes process more efficiently.

```
1 // Store the simulated sequences;
  std::vector<Sequence> sequences;
3 // Simulate 10 events for each sequence
  unsigned n = 2000;
5 // Simulate 2 sequences
  unsigned num_sequences = 10;
7 hawkes.Simulate(n, num_sequences, sequences);
```

Code Snippet 9.2: Efficient Hawkes Simulation

Notice on line 7, we call the simulation method specific to the `PlainHawkes` class, which indicates that the efficient method is only useful for the plain Hawkes process. To simulate a general point process, we still need to use the `OgataThinning` class.

### 9.3.2 Fitting

We can then fit a new Hawkes process to the simulated sequences with the following simple code.

```
// Define a new Hawkes object, with parameters uninitialized;
PlainHawkes hawkes_new(num_params, dim, beta);
// The input argument sequences store the simulated sequences from the simulation of
    the previous step.
PlainHawkes::OPTION options;
// We choose the projected LBFGS algorithm for the optimization.
options.method = PlainHawkes::PLBFGS;
hawkes_new.fit(sequences, options);
```

Code Snippet 9.3: Fitting a Hawkes Process

In this case, the final estimated parameters are 0.1060, 0.1949, 0.5029, 0.4899, 0.5196, 0.4844 which are pretty close to the true parameters shown in Figure 9.2. We can also put the sparsity constraint over the mutual excitation matrix by using L1 regularization, and put L22 regularization over the base intensity parameters when the number of dimension is large but we only have limited number of observations with the following code:

```
PlainHawkes hawkes_new(num_params, dim, beta);
// The input argument sequences store the simulated sequences
PlainHawkes::OPTION options;
// We choose the projected LBFGS algorithm for the optimization.
options.method = PlainHawkes::PLBFGS;
// Use L22 regularization over the base intensity
options.base_intensity_regularizer = PlainHawkes::L22;
// Use the L1 regularization over the mutual excitation matrix
options.excitation_regularizer = PlainHawkes::L1;
// Set the regularization coefficient for the base intensity
options.coefficients[LAMBDA] = 10;
// Set the regularization coefficient for the mutual excitation matrix
options.coefficients[BETA] = 70;
hawkes_new.fit(sequences, options);
```

Code Snippet 9.4: Fitting a Sparse Hawkes Process

We can also put the low-rank constraint over the mutual excitation matrix by using the nuclear norm regularization as the following.

```
// Define a new Hawkes object, with parameters uninitialized;
PlainHawkes hawkes_new(num_params, dim, beta);
// The input argument sequences store the simulated sequences from the simulation of
    the previous step.
PlainHawkes::OPTION options;
// We choose the projected LBFGS algorithm for the optimization.
options.method = PlainHawkes::PLBFGS;
// Use L22 regularization over the base intensity
options.base_intensity_regularizer = PlainHawkes::L22;
// Use the L1 regularization over the mutual excitation matrix
options.excitation_regularizer = PlainHawkes::NUCLEAR;
// Set the regularization coefficient for the base intensity
options.coefficients[LAMBDA] = 1;
// Set the regularization coefficient for the mutual excitation matrix
options.coefficients[BETA] = 1;
hawkes_new.fit(sequences, options);
```

Code Snippet 9.5: Fitting a Hawkes Process with Low-Rank constraint

### 9.3.3 Visualization

We can visualize the simulated events and the intensity functions of our Hawkes process with the following code.

```
// Initialize a Hawkes process;
PlainHawkes hawkes(num_params, dim, beta);
hawkes.SetParameters(params);
// Simulate one sequence until time 10
std::vector<double> vec_T(1, 10);
// Store the simulated sequences;
std::vector<Sequence> sequences;
// Initialize the simulator;
OgataThinning ot(dim);
ot.Simulate(hawkes, vec_T, sequences);
// Plot the intensity function using one sequence of events
hawkes.PlotIntensityFunction(sequences[0]);
```

Code Snippet 9.6: Visualizing a 2-d Hawkes Process

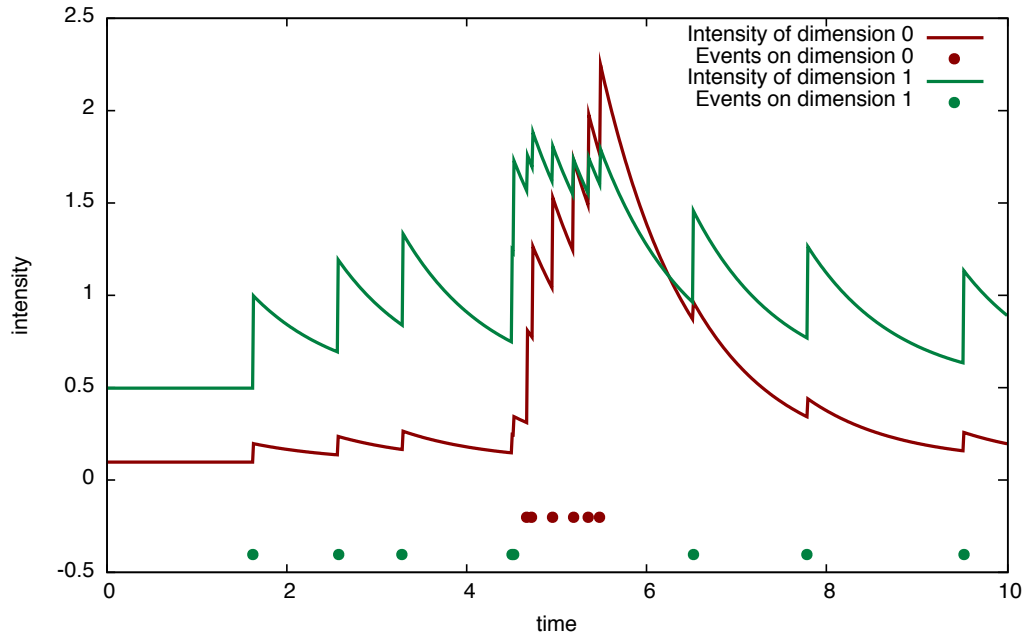



Figure 9.3: Visualizing a 2-d Hawkes process with `PtPack`.

The `PlotIntensityFunction` method on the last line plot the simulated events as well as the respective intensity functions, which is further shown in Figure 9.3 where the events and the respective intensity function on each dimension is marked with different colors.

### 9.3.4 Customized Triggering Kernels

The `PlainHawkes` class encapsulates the implementation of standard Hawkes Process [72] where the triggering kernel is an exponential function which assumes that the influence of a past event to triggering a new event in the future decays exponentially along time. However, in practice, we may want to design our own kernel functions to represent different triggering effects of interests, such as the periodicity. Our `PtPack` library allows users to define and plugin their own triggering kernel functions into a generalized Hawkes process by simply implementing a `TriggeringKernel` interface. By default, `PtPack` provides four different kernel functions: linear, sine, power-law and Rayleigh. The following code simulates a 1-d Hawkes process with the

sine triggering kernel and plots it in Figure 9.4.

```
// set the number of dimensions (dim) to be 1;
unsigned dim = 1, num_params = dim * (dim + 1);
// Initialize model parameters as a single vector;
Eigen::VectorXd params(num_params);
params << 0.5, 0.1;
// Initialize the triggering kernels
std::vector<std::vector<TriggeringKernel*> > triggeringkernels(dim, std::vector<
    TriggeringKernel*>(dim, NULL));
for(unsigned m = 0; m < dim; ++ m)
{
  for(unsigned n = 0; n < dim; ++ n)
  {
    // pairwise sine triggering kernel
    triggeringkernels[m][n] = new SineKernel();
  }
}
// Initialize a Generalized Hawkes process;
HawkesGeneralKernel hawkes(num_params, dim, triggeringkernels);
hawkes.SetParameters(params);
// Simulate one sequence with 50 points
OgataThinning ot(dim);
std::vector<Sequence> sequences;
ot.Simulate(hawkes, 50, 1, sequences);
// Plot the intensity function using one sequence of events
hawkes.PlotIntensityFunction(sequences[0]);
```

Code Snippet 9.7: Visualizing a 1-d Generalized Hawkes Process with a Sine Triggering Kernel

## 9.4 Summary

`PtPack` contributes to the machine learning community by providing efficient implementations of fitting and assessing high-dimensional temporal point processes over large-scale asynchronous event data. Compared to the existing libraries, it supports the following functionalities:

- Building, simulating, fitting, diagnosing and visualizing customized high-dimensional

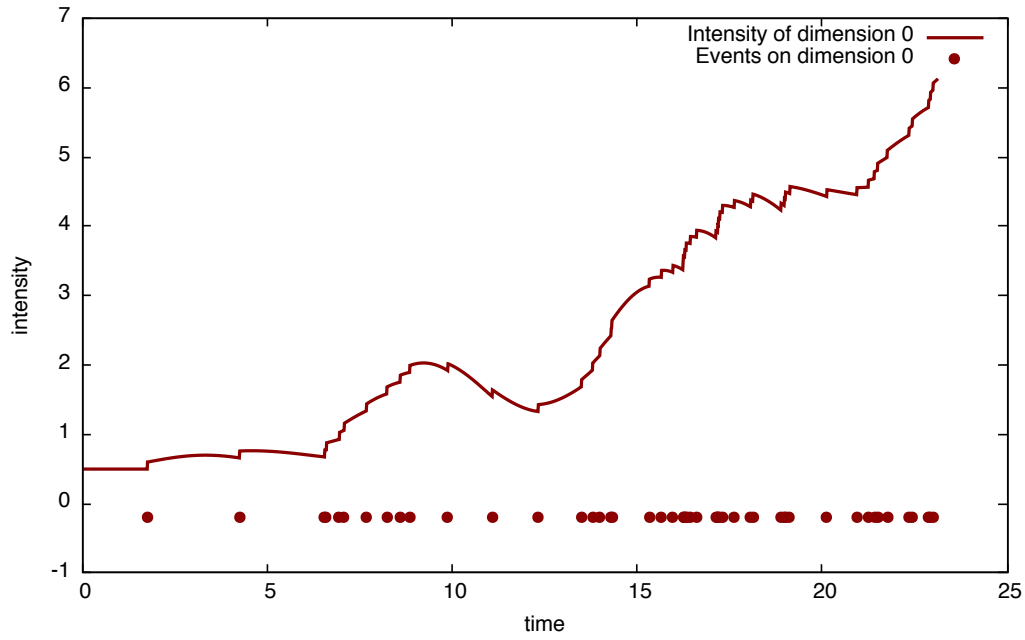



Figure 9.4: Visualizing a 1-d Hawkes process with `PtPack` using a sine triggering kernel.

temporal point processes in addition to the standard Poisson, Hawkes, Terminating point processes *etc.*

- Efficient implementation of state-of-the-arts learning algorithms to support different constraints, including: L1, L2, Group-lasso, Low-rank, over the interdependency structure of the dimensions.

- Large-scale inference for the expected number of events and average intensity measure for making time-sensitive decisions in the future.

The open source nature of the package may encourage other point process experts and machine learning practitioners to bring in new algorithms and features to `PtPack`. We will also keep updating and maintaining the package. More latest documents and codes are available at https://github.com/dunan/MultiVariatePointProcess.

# CHAPTER X

# CONCLUSIONS

The ubiquity and the increasing accessibility of temporal event data constantly produced from human activities, social networks, and other systems of interacting entities provide researchers new opportunities to analytically study the underlying dynamics that might be common across different domains to be responsible for these seemingly 'random' events. Effective modeling such latent dynamics not only allows us to better understand how different types of events might occur in specific settings, but also enables us to design and impose rules and incentives to further improve the underlying supporting systems and influence the behavior of each involved entity.

Centered on addressing the general '*who will do what by when and where?*' question, this dissertation provides a unified framework comprising of predictive models, efficient learning methods, and scalable inference algorithms based on multivariate point process. This framework effectively represents, captures, and optimizes the asynchronously and interdependently event data which often require much more sophisticated analyzing methods far beyond the existing approaches based on independent and identically distributed data models in literature. The design of the framework adheres to the following three steps: (1) we first start with developing predictive models to capture different temporal and structural characteristics of the asynchronous high-dimensional event data; (2) by exploiting the specific structural properties of each model's formulation, we propose efficient learning algorithms to train the models from massive datasets; and finally, (3) based on the learned models, we further develop scalable algorithms for large-scale inferences in order to help

people make time-sensitive decisions. In particular, we have made the following contributions:

**Models:**

- We propose nonparametric and topic-modulated multivariate terminating point processes to capture the dynamics of continuous-time information diffusions, which can better uncover the latent diffusion network structure.
- We develop the low-rank Hawkes process to model the recurrent temporal interactions between users and items, which is able to capture the context-aware preferences of users.
- In order to learn a general representation of the dependency over the history of both event timings and markers, we present the recurrent temporal point process, which innovatively connects the recurrent neural networks with point processes, which leads to significantly better prediction performance for both event type and timing.
- We propose the Dirichlet Point Process, which establishes a previously unexplored connection between Bayesian Nonparametrics and temporal point processes to combine the modeling of event data with other type of information, and allows the dimensions of the process to automatically grow so as to adapt the increasing complexity of the data streams.

**Learning:**

- We develop a robust structural learning algorithm via group lasso, which is able to efficiently uncover heterogeneous interdependency relations specified via vectorized parameters among the dimensions and can be easily parallelized in a distributed setting.

- We propose an efficient matrix rank minimization algorithm, which elegantly inherits the advantages from both the proximal methods and the conditional gradient methods to solve the matrix rank minimization problem under different constraints.

- We also develop an online Bayesian inference algorithm for inferring the latent cluster assignment and updating the respective model parameters based on both temporal and textual information, which achieves almost constant processing time per document in a stream of one million documents.

**Inference:**

- Based on the learned information diffusion models, we develop the first scalable influence estimation algorithm in continuous-time diffusion networks with general heterogeneous pairwise transmission functions. When used as a subroutine in a greedy influence maximization algorithm, our method can find the near-optimal solution with performance guarantees and can scale up to real networks of one million nodes. Furthermore, we are the first to rigorously evaluate the solution quality against real independent testing data for influence maximization, which further verifies that the algorithm can indeed generate greater influence compared with other state-of-the-arts.

- We have proposed the time-sensitive recommendation algorithm, which not only can recommend the most relevant item specific to a given moment, but also can predict the next returning time for a user to a designated service, which has great potentials to improve the Ad bidding strategies for web companies.

- Based on the recurrent user-activity model, we develop an analytic solution to calculate the expected overall network activity given the exogenous inputs from each individual user. Then, we further propose a set of convex formulations to

boost and shape user activities satisfying different budget constraints of interest. We show that our framework can provide more fine-grained control of network activity in a time-sensitive fashion.

**Open Source Software:** We provide efficient C++ implementations of building, fitting, and assessing high-dimensional temporal point processes via the open source package `PtPack`. Both researchers and practitioners in academia and industry can benefit from this open source package. For researchers, `PtPack` serves as good baselines for comparing with their new models to study the information dynamics in social networks for example. For practitioners in industry, they can tailor `PtPack` to best fit the requirements of their applications.

We believe these contributions can open several interesting research directions towards solving the general question of predicting '*who will do what by when and where?*' in the future. The long term goal of our research is to build increasingly accurate predictive models of natural and artificial systems in our world to predict future events, and to design better systems for delivering more intelligent and personalized services to the users. More specifically, our research will continue to focus on the following directions:

**Personalized Real-time Recommendation.** We keep investigating the connection between recurrent neural networks and temporal point processes. In addition to train a global model that captures the general sequential patterns on the population level, we will also take the local sequential history of each individual entity into considerations. Thus, given the current state and time of a user, the prediction of his (her) next state and time will depend not only on the outputs from the global models but also on the predictions of local models built from his (her) own trajectories. Based on the final prediction results and the users' current contexts, we might provide better recommendations.

**Dynamic Knowledge Graph.** Our learning algorithms for uncovering the latent interdependency structure among interacting entities can be applied to many other domains. For instance, knowledge graph is a powerful network structure connecting related but different types of concepts and entities to each other. Based on the news reports and mass media information, can we also infer the dynamics over the static knowledge graph structure? Can predict what kind of interactions (collaborations vs. conflicts) will happen by when and where given some regional events that currently occur to some of the entities in the graph?

**Adaptive Decision Making.** Based on the medical records of patients, we may be able to track the temporal dynamics that determine how quickly one type of disease may trigger the progression of other related complications, infections and even new diseases in the future. Given the current medical diagnosis of a patient, we want to adaptively decide: when should the patient to come back for the next observation? what is the best time to treat the patient? and what type of treatment is the most appropriate?

**More Efficient Learning Algorithms.** Normally, fitting a particular point process involves finding the maximum likelihood estimation for the model parameters. Depending on the formulation of the conditional intensity function, the optimization problem may be either convex (*e.g.* Hawkes process) or non-convex (*e.g.* Recurrent temporal point process). If the problem is convex, can we develop more efficient online, stochastic, or batch algorithms by exploiting the specific properties of the problem? If the problem is non-convex, can we also provide algorithms that guarantee to find the optimal solution under specific constraints for the problem? The development of these methods will inevitably make the point processes to be applied to an even broader range of applications in the future.

# APPENDIX A

# THEOREM PROOFS IN CHAPTER 4

## A.1 Heterogeneous Transmission Functions

We denote the waiting time distribution, or transmission function, along a directed edge of $\mathcal{G}$ as $f_{ji}(t_i|t_j)$. Formally, the transmission function $f_{ji}(t_i|t_j)$ for directed edge $j \to i$ is the conditional density of node $i$ getting infected at time $t_i$ given that node $j$ was infected at time $t_j$. We assume it is shift invariant, *i.e.*, $f_{ji}(t_i|t_j) = f_{ji}(t_i - t_j) = f_{ji}(\tau_{ji})$, where $\tau_{ji} := t_i - t_j$, and it takes positive values when $\tau_{ji} \geq 0$, and the value of zero otherwise.

In most previous work, simple parametric transmission functions such as the exponential distribution $\alpha_{ji} \exp(-\alpha_{ji}\tau_{ji})$, and the Rayleigh distribution $\alpha_{ji}\tau \exp(-\alpha_{ji}\tau_{ji}^2/2)$ have been used [56]. However, in many real world scenarios, information transmission between pairs of nodes can be heterogeneous and the waiting times can obey distributions that dramatically differ from these simple models. For instance, in viral marketing, active consumers could update their status instantly, while an inactive user may just log in and respond once a day. As a result, the transmission function between an active user and his friends can be quite different from that between an inactive user and his friends. As an attempt to model these complex scenarios, nonparametric transmission functions have been recently considered [44]. In such approach, the relationship between the survival function, the conditional intensity function or hazard, and the transmission function is exploited. In particular, the survival function is defined as $S_{ji}(\tau_{ji}) := 1 - \int_0^{\tau_{ji}} f_{ji}(\tau')d\tau'$ and the hazard function is defined as $h_{ji}(\tau_{ji}) := f_{ji}(\tau_{ji})/S_{ji}(\tau_{ji})$. Then, it is a well-known result in survival

theory that $S_{ji}(\tau_{ji}) = \exp\left(-\int_0^{\tau_{ji}} h_{ji}(\tau')d\tau'\right)$ and $f_{ji}(\tau_{ji}) = h_{ji}(\tau_{ji})S_{ji}(\tau_{ji})$. The advantage of using the conditional intensity function is that we do not need to explicitly enforce "the integral equals 1" constraint for the conditional density $f_{ji}$. Instead, we just need to ensure $h_{ji} \geq 0$. This facilitates nonparametric modeling of the transmission function. For instance, we can define the conditional intensity function as a positive combination of $n$ positive kernel functions $k$,

$$h_{ji}(\tau) = \sum\nolimits_{l=1}^{n} \alpha_l k(\tau_l, \tau), \text{ if } \tau > 0, \text{ and 0 otherwise.}$$

A common choice of the kernel function is the Gaussian RBF kernel $k(\tau', \tau) = \exp(-\left\|\tau - \tau'\right\|^2 / 2s^2)$. Nonparametric transmission functions significantly improve modeling of real world diffusion, as is shown in [44].

## A.2 A Graphical Model Perspective

Now, we look at the independent cascade model from the perspective of graphical models, where the collection of random variables includes the infection times $t_i$ of the nodes. Although the original contact graph $\mathcal{G}$ can contain directed loops, each diffusion process (or a cascade) induces a directed acyclic graph (DAG). For those cascades consistent with a particular DAG, we can model the joint density of $t_i$ using a directed graphical model:

$$p\left(\{t_i\}_{i\in\mathcal{V}}\right) = \prod\nolimits_{i\in\mathcal{V}} p\left(t_i | \{t_j\}_{j\in\pi_i}\right), \tag{A.1}$$

where each $\pi_i$ denotes the collection of parents of node $i$ in the induced DAG, and each term $p(t_i|\{t_j\}_{j\in\pi_i})$ corresponds to a conditional density of $t_j$ given the infection times of the parents of node $i$. This is true because given the infection times of node $i$'s parents, $t_i$ is independent of other infection times, satisfying the local Markov property of a directed graphical model. We note that the independent cascade model only specifies explicitly the pairwise transmission functions for each directed edge, but does not directly define the conditional density $p(t_i|\{t_j\}_{j\in\pi_i})$.

However, these conditional densities can be derived from the pairwise transmission functions based on the Independent-Infection property [56]:

$$p\left(t_i|\{t_j\}_{j\in\pi_i}\right) = \sum\nolimits_{j\in\pi_i} h_{ji}(t_i|t_j) \prod\nolimits_{l\in\pi_i} S(t_i|t_l), \tag{A.2}$$

which is the sum of the likelihoods that node $i$ is infected by each parent node $j$. More precisely, each term in the summation can be interpreted as the instantaneous risk of node $i$ being infected at $t_i$ by node $j$ given that it has survived the infection of all parent nodes until time $t_i$.

Perhaps surprisingly, the factorization in Equation (A.1) is the same factorization that can be used for an arbitrary induced DAG consistent with the contact network $\mathcal{G}$. In this case, we only need to replace the definition of $\pi_i$ (the parent of node $i$ in the DAG) to the set of neighbors of node $i$ with an edge pointing to node $i$ in $\mathcal{G}$. This is not immediately obvious from Equation (A.1), since the contact network $\mathcal{G}$ can contain directed loops which may be in conflict with the conditional independence semantics of directed graphical models. The reason it is possible to do so is as follows: Any fixed set of infection times, $t_1, \ldots, t_d$, induces an ordering of the infection times. If $t_i \leq t_j$ for an edge $j \to i$ in $\mathcal{G}$, $h_{ji}(t_i|t_j) = 0$, and the corresponding term in Equation (A.2) is zeroed out, making the conditional density consistent with the semantics of directed graphical models.

Based on the joint density of the infection times in Equation (A.1), we can perform various inference and learning tasks. For instance, previous work has used Equation (A.1) for learning the parameters of the independent cascade model [44, 56, 99]. However, this may not be the most convenient form for addressing other inference problems, including the influence estimation problem in the next section. To this end, we propose an alternative view.

Instead of directly modeling the infection times $t_i$, we can focus on the collection of mutually independent random transmission times $\tau_{ji} = t_i - t_j$. In this case, the

joint density of the collection of transmission times $\tau_{ji}$ is fully factorized

$$p\left(\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}\right) = \prod\nolimits_{(j,i)\in\mathcal{E}} f_{ji}(\tau_{ji}),$$

where $\mathcal{E}$ denotes the set of edges in the contact network $\mathcal{G}$ — switching from the earlier node-centric view to the now edge-centric view. Based on the Shortest-Path property of the independent cascade model, variable $t_i$ can be viewed as a transformation from the collection of variables $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$. More specifically, let $\mathcal{Q}_i$ be the collection of directed paths in $\mathcal{G}$ from the source nodes to node $i$, where each path $q \in \mathcal{Q}_i$ contains a sequence of directed edges $(j, l)$, and assuming all source nodes are infected at zero time, then we obtain variable $t_i$ via

$$t_i = g_i\left(\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}\right) := \min_{q\in\mathcal{Q}_i} \sum\nolimits_{(j,l)\in q} \tau_{jl}, \tag{A.3}$$

where $g_i(\cdot)$ is the transformation.

Importantly, we can now compute the probability of infection of node $i$ at $t_i$ using the set of variables $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$:

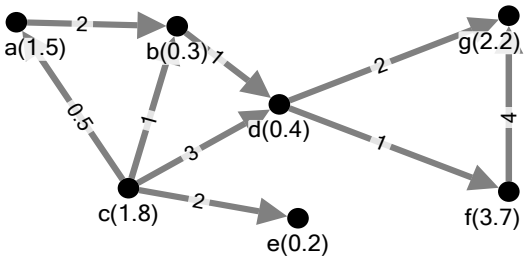
$$\Pr\{t_i \le T\} = \Pr\left\{g_i\left(\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}\right) \le T\right\}. \tag{A.4}$$

The significance of the relation is that it allows us to transform a problem involving a sequence of dependent variables $\{t_i\}_{i\in\mathcal{V}}$ to one with independent variables $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$. Furthermore, the two problems are connected via the shortest path algorithm in weighted directed graph, a standard well studied operation in graph analysis.

## A.3 Naive Sampling Algorithm

The graphical model perspective described in Section 4.2 and Appendix A.2 suggests a naive sampling (NS) algorithm for approximating $\sigma(\mathcal{A}, T)$:

1. Draw $n$ samples, $\left\{\left\{\tau_{ji}^l\right\}_{(j,i)\in\mathcal{E}}\right\}_{l=1}^{n}$, *i.i.d.* from the waiting time product distribution $\prod_{(j,i)\in\mathcal{E}} f_{ji}(\tau_{ji})$;




- Node labeling:
  $e(0.2) < b(0.3) < d(0.4) < a(1.5) < c(1.8) < g(2.2) < f(3.7)$
- Neighborhoods:
  $\mathcal{N}(c, 2) = \{a, b, c, e\}; \mathcal{N}(c, 3) = \{a, b, c, d, e, f\};$
- Least-label list:
  $r_*(c) : (2, 0.2), (1, 0.3), (0.5, 1.5), (0, 1.8)$
- Query: $r_*(c, 0.8) = r(a) = 1.5$

Figure A.1: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, edge weights $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$, and node labeling $\{r_i\}_{i\in\mathcal{V}}$ with the associated output from Algorithm A.1.

2. For each sample $\left\{\tau_{ji}^l\right\}_{(j,i)\in\mathcal{E}}$ and for each node $i$, find the shortest path from source nodes to node $i$; count the number of nodes with $g_i\left(\left\{\tau_{ji}^l\right\}_{(j,i)\in\mathcal{E}}\right) \leq T$;
3. Average the counts across $n$ samples.

Although the naive sampling algorithm can handle arbitrary transmission function, it is not scalable to networks with millions of nodes. We need to compute the shortest path for each node and each sample, which results in a computational complexity of $O(n|\mathcal{E}| + n|\mathcal{V}|\log|\mathcal{V}|)$ for a single source node. The problem is even more pressing in the influence maximization problem, where we need to estimate the influence of source nodes at different location and with increasing number of source nodes. To do this, the algorithm needs to be repeated, adding a multiplicative factor of $C|\mathcal{V}|$ to the computational complexity ($C$ is the number of nodes to select). Then, the algorithm becomes quadratic in the network size. When the network size is in the order of thousands and millions, typical in modern social network analysis, the naive sampling algorithm become prohibitively expensive. Additionally, we may need to draw thousands of samples ($n$ is large), further making the algorithm impractical for large scale problems.

## A.4 Least Label List

The notation "argsort$((r_1, \ldots, r_{|\mathcal{V}|}), \text{ascend})$" in line 2 of Algorithm A.1 means that we sort the collection of random labels in ascending order and return the argument

**Algorithm A.1:** Least Label List

**Input**: a reversed directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with edge weights $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$, a node labeling $\{r_i\}_{i\in\mathcal{V}}$

**Output**: A list $r_*(s)$ for each $s \in \mathcal{V}$

**for** *each* $s \in \mathcal{V}$ **do** $d_s \leftarrow \infty, r_*(s) \leftarrow \emptyset$;
**for** $i$ *in argsort*$((r_1, \ldots, r_{|\mathcal{V}|}), \textit{ascend})$ **do**
    empty heap H $\leftarrow \emptyset$;
    set all nodes except $i$ as unvisited;
    push $(0, i)$ into heap H;
    **while** H $\neq \emptyset$ **do**
        pop $(d_*, s)$ with the minimum $d_*$ from H;
        add $(d_*, r_i)$ to the end of list $r_*(s)$;
        $d_s \leftarrow d^*$;
        **for** *each unvisited out-neighbor* $j$ *of* $s$ **do**
            set $j$ as visited;
            **if** $(d, j)$ *in heap* H **then**
                Pop $(d, j)$ from heap H;
                Push $(\min\{d, d_* + \tau_{js}\}, j)$ into heap H;
            **else if** $d_* + \tau_{js} < d_j$ **then**
                Push $(d_* + \tau_{js}, j)$ into heap H;

of the sort as an ordered list. Figure A.1 shows an example of the Least-Label-List. The nodes from $a$ to $g$ are assigned to exponentially distributed labels with mean one shown in each parentheses. Given a query distance 0.8 for node $c$, we can binary-search its Least-label-list $r_*(c)$ to find that node $a$ belongs to this range with the smallest label $r(a) = 1.5$.

## A.5 Proof of Theorem 4.1

**Theorem 4.1** *Sample the following number of sets of random transmission times*

$$n \geqslant \frac{C\Lambda}{\epsilon^2} \log\left(\frac{2|\mathcal{V}|}{\delta}\right)$$

*where* $\Lambda := \max_{\mathcal{A}:|\mathcal{A}|\leq C} 2\sigma(\mathcal{A}, T)^2/(m-2) + 2Var(|\mathcal{N}(\mathcal{A}, T)|)(m-1)/(m-2) + 2a\epsilon/3$, $|\mathcal{N}(\mathcal{A}, T)| \leqslant a$, *and for each set of random transmission times, sample* $m$ *set of random labels. Then we can guarantee that*

$$|\widehat{\sigma}(\mathcal{A}, T) - \sigma(\mathcal{A}, T)| \leqslant \epsilon$$

*simultaneously for all $\mathcal{A}$ with $|\mathcal{A}| \leqslant C$, with probability at least $1-\delta$.*

**Proof** Let $S_\tau := |\mathcal{N}(\mathcal{A}, T)|$ for a fixed set of $\{\tau_{ji}\}$ and then $\sigma(\mathcal{A}, T) = \mathbb{E}_\tau[S_\tau]$. The randomized algorithm with $m$ randomizations produces an unbiased estimator $\widehat{S}_\tau = (m-1)/(\sum_{u=1}^{m} r_*^u)$ for $S_\tau$, *i.e.*, $\mathbb{E}_{r|\tau}[\widehat{S}_\tau] = S_\tau$, with variance $\mathbb{E}_{r|\tau}[(\widehat{S}_\tau - S_\tau)^2] = S_\tau^2/(m-2)$.

Then $\widehat{S}_\tau$ is also an unbiased estimator for $\sigma(\mathcal{A}, T)$, since $\mathbb{E}_{\tau,r}[\widehat{S}_\tau] = \mathbb{E}_\tau \mathbb{E}_{r|\tau}[\widehat{S}_\tau] = \mathbb{E}_\tau[S_\tau] = \sigma(\mathcal{A}, T)$. Its variance is

$$\begin{aligned} Var(\widehat{S}_\tau) &:= \mathbb{E}_{\tau,r}[(\widehat{S}_\tau - \sigma(\mathcal{A}, T))^2] = \mathbb{E}_{\tau,r}[(\widehat{S}_\tau - S_\tau + S_\tau - \sigma(\mathcal{A}, T))^2] \\ &= \mathbb{E}_{\tau,r}[(\widehat{S}_\tau - S_\tau)^2] + 2\,\mathbb{E}_{\tau,r}[(\widehat{S}_\tau - S_\tau)(S_\tau - \sigma(\mathcal{A}, T))] + \mathbb{E}_{\tau,r}[(S_\tau - \sigma(\mathcal{A}, T))^2] \\ &= \mathbb{E}_\tau[S_\tau^2/(m-2)] + 0 + Var(S_\tau) \\ &= \sigma(\mathcal{A}, T)^2/(m-2) + Var(S_\tau)(m-1)/(m-2) \end{aligned}$$

Then using Bernstein's inequality, we have, for our final estimator $\widehat{\sigma}(\mathcal{A}, T) = \frac{1}{n}\sum_{l=1}^{n} \widehat{S}_{\tau^l}$, that

$$\Pr\left\{|\widehat{\sigma}(\mathcal{A}, T) - \sigma(\mathcal{A}, T)| \geqslant \epsilon\right\} \leqslant 2\exp\left(-\frac{n\epsilon^2}{2Var(\widehat{S}_\tau) + 2a\epsilon/3}\right) \tag{A.5}$$

where $\widehat{S}_\tau < a \leqslant |\mathcal{V}|$.

Setting the right hand side of relation (A.5) to $\delta$, we have that, with probability $1-\delta$, sampling the following number sets of random transmission times

$$\begin{aligned} n &\geqslant \frac{2Var(\widehat{S}_\tau) + 2a\epsilon/3}{\epsilon^2} \log\left(\frac{2}{\delta}\right) \\ &= \frac{2\sigma(\mathcal{A}, T)^2/(m-2) + 2Var(S_\tau)(m-1)/(m-2) + 2a\epsilon/3}{\epsilon^2} \log\left(\frac{2}{\delta}\right) \end{aligned}$$

we can guarantee that our estimator to have error $|\widehat{\sigma}(\mathcal{A}, T) - \sigma(\mathcal{A}, T)| \leqslant \epsilon$.

If we want to insure that $|\widehat{\sigma}(\mathcal{A}, T) - \sigma(\mathcal{A}, T)| \leqslant \epsilon$ simultaneously hold for all $\mathcal{A}$ such that $|\mathcal{A}| \leqslant C \ll |\mathcal{V}|$, we can first use union bound with relation (A.5). In this case, we have that, with probability $1-\delta$, sampling the following number sets of

random transmission times

$$n \geqslant \frac{C\Lambda}{\epsilon^2} \log\left(\frac{2|\mathcal{V}|}{\delta}\right)$$

we can guarantee that our estimator to have error $|\widehat{\sigma}(\mathcal{A}, T) - \sigma(\mathcal{A}, T)| \leqslant \epsilon$ for all $\mathcal{A}$ with $|\mathcal{A}| \leqslant C$. Note that we have define the constant $\Lambda := \max_{\mathcal{A}:|\mathcal{A}|\leq C} 2\sigma(\mathcal{A}, T)^2/(m-2) + 2Var(S_\tau)(m-1)/(m-2) + 2a\epsilon/3$. ■

# APPENDIX B

# THEOREM PROOFS IN CHAPTER 5

## B.1 Proof of Theorem 5.2

**Theorem 5.2** *With the condition $\rho \geq \rho^*$, the optimal value $\widehat{\mathrm{OPT}}$ of the problem 5.5 coincides with the optimal value* OPT *in the problem B.1 of interest, where $\rho^*$ is a problem dependent threshold.*

**Proof** *We start by rewriting formulation 5.4 to the equivalent form:*

$$\min -\frac{1}{|\mathcal{O}|}\sum_{\mathcal{T}^{u,i}\in\mathcal{O}} \ell\left(\mathcal{T}^{u,i}|\boldsymbol{\Lambda}_0, \boldsymbol{A}\right) + \lambda\|\boldsymbol{Z}_1\|_* + \beta\|\boldsymbol{Z}_2\|_*,$$
$$\boldsymbol{\Lambda}_0, \boldsymbol{A} \geq \mathbf{0}, \boldsymbol{\Lambda}_0 = \boldsymbol{Z}_1, \boldsymbol{A} = \boldsymbol{Z}_2. \tag{B.1}$$

*We can observe that the optimal solution of B.1 is a feasible solution of 5.5 with the same objective function value, so it is evident that $\widehat{\mathrm{OPT}} \leq \mathrm{OPT}$. On the other hand, suppose $(\boldsymbol{\Lambda}_0^*, \boldsymbol{A}^*, \boldsymbol{Z}_1^*, \boldsymbol{Z}_2^*)$ is an optimal solution of 5.5 with $\boldsymbol{\Lambda}_0^* \neq \boldsymbol{Z}_1^*, \boldsymbol{A}^* \neq \boldsymbol{Z}_2^*$ in general. Since $\boldsymbol{\Lambda}_0^*, \boldsymbol{A}^* \geq \mathbf{0}$, they are also feasible for B.1, so we can find a $\rho'$ such that $\lambda\|\boldsymbol{Z}_1^*\|_* + \beta\|\boldsymbol{Z}_2^*\|_* + \rho'\|\boldsymbol{\Lambda}_0^* - \boldsymbol{Z}_1^*\|_F^2 + \rho'\|\boldsymbol{A}^* - \boldsymbol{Z}_2^*\|_F^2 \geq \lambda\|\boldsymbol{\Lambda}_0^*\|_* + \beta\|\boldsymbol{A}^*\|_*$. Therefore, under the condition that*

$$\rho \geq \rho^* = \max\left\{\frac{\lambda\left(\|\boldsymbol{\Lambda}_0^*\|_* - \|\boldsymbol{Z}_1^*\|_*\right) + \beta\left(\|\boldsymbol{A}^*\|_* - \|\boldsymbol{Z}_2^*\|_*\right)}{\|\boldsymbol{\Lambda}_0^* - \boldsymbol{Z}_1^*\|_F^2 + \|\boldsymbol{A}^* - \boldsymbol{Z}_2^*\|_F^2}\right\}, \tag{B.2}$$

*we have $\widehat{\mathrm{OPT}} \geq -\frac{1}{|\mathcal{O}|}\sum_{\mathcal{T}^{u,i}\in\mathcal{O}} \ell\left(\mathcal{T}^{u,i}|\boldsymbol{\Lambda}_0^*, \boldsymbol{A}^*\right) + \lambda\|\boldsymbol{\Lambda}_0^*\|_* + \beta\|\boldsymbol{A}^*\|_* \geq \mathrm{OPT}$ and readily arrive at the theorem.* ■

## B.2 Proof of Theorem 5.3

**Theorem 5.3** *Let $\{\boldsymbol{Y}^k\}$ be the sequence generated by Algorithm 5.1, $\delta^k = 2/(k+1)$, and $\eta^k = (\delta^k)^{-1}/L$, $D_1$ and $D_2$ some problem dependent constants. Then for $k \geq 1$, we have*

$$F(\boldsymbol{Y}^k) - F(\boldsymbol{X}^*) \leq \frac{4LD_1}{k(k+1)} + \frac{2LD_2}{k+1}. \tag{B.3}$$

**Proof** *Consider the following general optimization problem*

$$\min_{\boldsymbol{X}\in\Omega} F(\boldsymbol{X}) := f(\boldsymbol{X}_1; \boldsymbol{X}_2) + \Psi(\boldsymbol{X}_2), \tag{B.4}$$

*where $\boldsymbol{X} = [\boldsymbol{X}_1; \boldsymbol{X}_2]$, $\Omega = \Omega_1 \times \Omega_2$, $f$ is $L$-smooth and convex, and $\Psi(\cdot)$ is convex. Let $\delta^k = \frac{1}{k+2}$ and $\eta^k = (\delta^k)^{-1}/L$. First Note that $\boldsymbol{Y}^k - \boldsymbol{U}^{k-1} = \delta^k(\boldsymbol{X}^k - \boldsymbol{X}^{k-1})$. By the smoothness of $f$ where $f(y) \leq f(x) + \langle f'(x), y - x\rangle + \frac{L}{2}\|y - x\|^2$, we have*

$$\begin{aligned}
f(\boldsymbol{Y}^k) &\leq f(\boldsymbol{U}^{k-1}) + \nabla f(\boldsymbol{U}^{k-1})^\top \left(\boldsymbol{Y}^k - \boldsymbol{U}^{k-1}\right) + \tfrac{L}{2}\delta_k^2\|\boldsymbol{X}_k - \boldsymbol{X}_{k-1}\|^2 \\
&\quad \left(\textit{by the definition of } \boldsymbol{Y}^k\right) \\
&= (1-\delta^k)\left(f(\boldsymbol{U}^{k-1}) + \nabla f(\boldsymbol{U}^{k-1})^\top \left(\boldsymbol{Y}^{k-1} - \boldsymbol{U}^{k-1}\right)\right) \\
&\quad +\delta^k\left(f(\boldsymbol{U}^{k-1}) + \nabla f(\boldsymbol{U}^{k-1})^\top \left(\boldsymbol{X}^k - \boldsymbol{U}^{k-1}\right)\right) + \tfrac{L}{2}\delta_k^2\|\boldsymbol{X}_k - \boldsymbol{X}_{k-1}\|^2 \\
&\quad (\textit{by the convexity of } f) \\
&\leq (1-\delta^k) f(\boldsymbol{Y}^{k-1}) + \delta^k\left(f(\boldsymbol{U}^{k-1}) + \nabla f(\boldsymbol{U}^{k-1})^\top \left(\boldsymbol{X}^k - \boldsymbol{U}^{k-1}\right)\right) \\
&\quad +\tfrac{L}{2}\delta_k^2\|\boldsymbol{X}_k - \boldsymbol{X}_{k-1}\|^2.
\end{aligned} \tag{B.5}$$

*Note the proximal mapping $\mathrm{Prox}_{x_0}(\xi) := \mathrm{argmin}_{x\in X}\{V(x, x_0) + \langle \xi, x\rangle\}$, where $V(x, x') = \omega(x) - \omega(x') - \langle \nabla\omega(x'), x - x'\rangle$ is the Bregman distance, and $\omega(x)$ is 1-strongly convex. For any $\boldsymbol{X}_1 \in \Omega_1$, we have the following well-known inequality [126]:*

$$\nabla_1 f(\boldsymbol{U}^{k-1})^T(\boldsymbol{X}_1^k - \boldsymbol{X}_1) \leq (\eta^k)^{-1}[V(\boldsymbol{X}_1, \boldsymbol{X}_1^{k-1}) - V(\boldsymbol{X}_1, \boldsymbol{X}_1^k) - V(\boldsymbol{X}_1^k, \boldsymbol{X}_1^{k-1})]. \tag{B.6}$$

*Besides, by our linear minimization oracle*

$$\mathrm{LMO}_\Psi(\nabla_2 f(\boldsymbol{U}^{k-1})) = \mathrm{argmin}\left\{\left\langle \nabla_2 f(\boldsymbol{U}^{k-1}), \boldsymbol{X}_2\right\rangle + \Psi(\boldsymbol{X}_2)\right\}, \tag{B.7}$$

*we have*

$$\nabla_2 f(\boldsymbol{U}^{k-1})^\top \boldsymbol{X}_2^k + \Psi(\boldsymbol{X}_2^k) \leq \nabla_2 f(\boldsymbol{U}^{k-1})^\top \boldsymbol{X}_2 + \Psi(\boldsymbol{X}_2). \tag{B.8}$$

*As a consequence,*

$$\begin{aligned}
& \delta^k \left( f(\boldsymbol{U}^{k-1}) + \nabla f(\boldsymbol{U}^{k-1})^\top \left( \boldsymbol{X}^k - \boldsymbol{U}^{k-1} \right) \right) \\
= \; & \delta^k \left( f(\boldsymbol{U}^{k-1}) + \nabla_1 f(\boldsymbol{U}^{k-1})^\top \left( \boldsymbol{X}_1^k - \boldsymbol{U}_1^{k-1} \right) + \nabla_2 f(\boldsymbol{U}^{k-1})^\top \left( \boldsymbol{X}_2^k - \boldsymbol{U}_2^{k-1} \right) \right) \\
& (\textit{by equation B.8}) \\
\leq \; & \delta^k (f(\boldsymbol{U}^{k-1}) + \nabla_1 f(\boldsymbol{U}^{k-1})^\top \left( \boldsymbol{X}_1^k - \boldsymbol{U}_1^{k-1} + \boldsymbol{X}_1^* - \boldsymbol{X}_1^* \right) \\
& + \nabla_2 f(\boldsymbol{U}^{k-1})^\top \boldsymbol{X}_2^* + \Psi(\boldsymbol{X}_2^*) - \Psi(\boldsymbol{X}_2^k) - \nabla_2 f(\boldsymbol{U}^{k-1})^\top \boldsymbol{U}_2^{k-1})) \\
\leq \; & \delta^k (f(\boldsymbol{U}^{k-1}) + \nabla_1 f(\boldsymbol{U}^{k-1})^\top \left( \boldsymbol{X}_1^* - \boldsymbol{U}_1^{k-1} \right) + \nabla_2 f(\boldsymbol{U}^{k-1})^\top (\boldsymbol{X}_2^* - \boldsymbol{U}_2^{k-1}) \\
& + \Psi(\boldsymbol{X}_2^*) + \nabla_1 f(\boldsymbol{U}^{k-1})^\top \left( \boldsymbol{X}_1^k - \boldsymbol{X}_1^* \right) - \Psi(\boldsymbol{X}_2^k)) \\
& (\textit{by the convexity of } f) \\
\leq \; & \delta^k F(\boldsymbol{X}^*) + \delta^k \nabla_1 f(\boldsymbol{U}^{k-1})^\top \left( \boldsymbol{X}_1^k - \boldsymbol{X}_1^* \right) - \delta^k \Psi(\boldsymbol{X}_2^k) \\
& (\textit{by equation B.6}) \\
\leq \; & \delta^k F(\boldsymbol{X}^*) + \delta^k (\eta^k)^{-1} (V(\boldsymbol{X}_1^*, \boldsymbol{X}_1^{k-1}) - V(\boldsymbol{X}_1^*, \boldsymbol{X}_1^k) - V(\boldsymbol{X}_1^k, \boldsymbol{X}_1^{k-1})) - \delta^k \Psi(\boldsymbol{X}_2^k) \\
& (\textit{by the definition of Bregman distance}) \\
\leq \; & \delta^k F(\boldsymbol{X}^*) + L(\delta^k)^2 (V(\boldsymbol{X}_1^*, \boldsymbol{X}_1^{k-1}) - V(\boldsymbol{X}_1^*, \boldsymbol{X}_1^k)) - \tfrac{L(\delta^k)^2}{2} \|\boldsymbol{X}_1^k - \boldsymbol{X}_1^{k-1}\|^2 \\
& - \delta^k \Psi(\boldsymbol{X}_2^k)
\end{aligned}$$

*Plugging into the previous inequality B.5, we end up with*

$$\begin{aligned}
f(\boldsymbol{Y}^k) \leq & (1 - \delta^k) f(\boldsymbol{Y}^{k-1}) + \delta^k F(\boldsymbol{X}^*) + L(\delta^k)^2 (V(\boldsymbol{X}_1^*, \boldsymbol{X}_1^{k-1}) - V(\boldsymbol{X}_1^*, \boldsymbol{X}_1^k)) \\
& + \frac{L(\delta^k)^2}{2} \|\boldsymbol{X}_2^k - \boldsymbol{X}_2^{k-1}\|^2 - \delta^k \Psi(\boldsymbol{X}_2^k),
\end{aligned} \tag{B.9}$$

*where we have used the fact* $\|\boldsymbol{X} = (\boldsymbol{X}_1; \boldsymbol{X}_2)\|^2 = \|\boldsymbol{X}_1\|^2 + \|\boldsymbol{X}_2\|^2$.

*Adding* $\Psi(\boldsymbol{Y}_2^k)$ *to the both sides, we have*

$$\begin{aligned} F(\boldsymbol{Y}^k) &\leq (1-\delta^k)F(\boldsymbol{Y}^{k-1}) + \delta^k F(\boldsymbol{X}^*) + L(\delta^k)^2(V(\boldsymbol{X}_1^*, \boldsymbol{X}_1^{k-1}) - V(\boldsymbol{X}_1^*, \boldsymbol{X}_1^k)) \\ &\quad + \tfrac{L(\delta^k)^2}{2}\|\boldsymbol{X}_2^k - \boldsymbol{X}_2^{k-1}\|^2 + \Psi(\boldsymbol{Y}_2^k) - \delta^k\Psi(\boldsymbol{X}_2^k) - (1-\delta^k)\Psi(\boldsymbol{Y}_2^{k-1}) \\ &\quad \big(\textit{by the convexity of } \Psi \textit{ and the definition of } \boldsymbol{Y}^k\big) \\ &\leq (1-\delta^k)F(\boldsymbol{Y}^{k-1}) + \delta^k F(\boldsymbol{X}^*) + L(\delta^k)^2(V(\boldsymbol{X}_1^*, \boldsymbol{X}_1^{k-1}) - V(\boldsymbol{X}_1^*, \boldsymbol{X}_1^k)) \\ &\quad + \tfrac{L(\delta^k)^2}{2}\|\boldsymbol{X}_2^k - \boldsymbol{X}_2^{k-1}\|^2. \end{aligned} \tag{B.10}$$

*Subtracting* $F(\boldsymbol{X}^*)$ *from both sides of the above inequality, we have*

$$\begin{aligned} F(\boldsymbol{Y}^k) - F(\boldsymbol{X}^*) \leq &(1-\delta^k)(F(\boldsymbol{Y}^{k-1}) - F(\boldsymbol{X}^*)) + L(\delta^k)^2(V(\boldsymbol{X}_1^*, \boldsymbol{X}_1^{k-1}) \\ &- V(\boldsymbol{X}_1^*, \boldsymbol{X}_1^k)) + \frac{L(\delta^k)^2}{2}\|\boldsymbol{X}_2^k - \boldsymbol{X}_2^{k-1}\|^2. \end{aligned} \tag{B.11}$$

*By the fact* $\delta^1 = 1$ *and invoking the Lemma 1 of [96], the above inequality implies*

$$F(\boldsymbol{Y}^k) - F(\boldsymbol{X}^*) \leq \frac{4L}{k(k+1)}\left(V(\boldsymbol{X}_1^*, \boldsymbol{X}_1^0) + \frac{1}{2}\sum_{i=1}^{k}\|\boldsymbol{X}_2^k - \boldsymbol{X}_2^{k-1}\|^2\right). \tag{B.12}$$

*Let* $D_1 = V(\boldsymbol{X}_1^*, \boldsymbol{X}_1^0) \geq 0$ *and* $D_2 = \max_{x,y\in\Omega_2}\|x-y\|^2$*, we have*

$$F(\boldsymbol{Y}^k) - F(\boldsymbol{X}^*) \leq \frac{4LD_1}{k(k+1)} + \frac{2LD_2}{k+1}. \tag{B.13}$$

■

# APPENDIX C

# THEOREM PROOFS IN CHAPTER 6

## C.1 Proof of Lemma 6.1

**Lemma 6.1** $\boldsymbol{\mu}^{(k)}(t) = \boldsymbol{G}^{(\star k)}(t)\,\boldsymbol{\lambda}^{(0)}$.

**Proof** *We will prove the lemma by induction. For generation* $k = 0$, $\boldsymbol{\mu}^{(0)}(t) = \mathbb{E}_{\mathcal{H}_t}[\boldsymbol{\lambda}^{(0)}] = \boldsymbol{G}^{(\star 0)}(t)\boldsymbol{\lambda}^{(0)}$. *Assume the relation holds for generation* $k$: $\boldsymbol{\mu}^{(k)}(t) = \boldsymbol{G}^{(\star k)}(t)\boldsymbol{\lambda}^{(0)}$. *Then for generation* $k{+}1$, *we have* $\boldsymbol{\mu}^{(k+1)}(t) = \mathbb{E}_{\mathcal{H}_t}[\int_0^t \boldsymbol{G}(t{-}s)\,d\boldsymbol{N}^{(k)}(s)] = \int_0^t \boldsymbol{G}(t-s)\,\mathbb{E}_{\mathcal{H}_t}[d\boldsymbol{N}^{(k)}(s)]$. *By definition* $\mathbb{E}_{\mathcal{H}_t}[d\boldsymbol{N}^{(k)}(s)] = \mathbb{E}_{\mathcal{H}_{s-}}[\mathbb{E}[d\boldsymbol{N}^{(k)}(s)|\mathcal{H}_{s-}]] = \mathbb{E}_{\mathcal{H}_{s-}}[\boldsymbol{\lambda}^{(k)}(s)\,ds] = \boldsymbol{\mu}^{(k)}(s)\,ds$, *then substitute it in and we have*

$$\boldsymbol{\mu}^{(k+1)}(t) = \int_0^t \boldsymbol{G}(t-s)\,\boldsymbol{G}^{(\star k)}(t)\,\boldsymbol{\lambda}^{(0)}\,ds = \boldsymbol{G}^{(\star k+1)}(t)\boldsymbol{\lambda}^{(0)},$$

*which completes the proof.* ■

## C.2 Proof of Lemma 6.2

**Lemma 6.2** $\widehat{\boldsymbol{G}}^{(\star k)}(z) = \int_0^\infty \boldsymbol{G}^{(\star k)}(t)\,dt = \frac{1}{z}\cdot\frac{\boldsymbol{A}^k}{(z+\omega)^k}$.

**Proof** *We will prove the result by induction on* $k$. *First, given our choice of exponential kernel,* $\boldsymbol{G}(t) = e^{-\omega t}\boldsymbol{A}$, *we have that* $\widehat{\boldsymbol{G}}(z) = \frac{1}{z+w}\boldsymbol{A}$. *Then for* $k = 0$, $\boldsymbol{G}^{(\star 0)}(t) = \boldsymbol{I}$ *and* $\widehat{\boldsymbol{I}}(z) = \int_0^\infty e^{-zt}\boldsymbol{I}\,dt = \frac{1}{z}\boldsymbol{I}$. *Now assume the result hold for a general* $k-1$, *then* $\widehat{\boldsymbol{G}}^{(\star k-1)}(z) = \frac{1}{z}\cdot\frac{\boldsymbol{A}^{k-1}}{(z+\omega)^{k-1}}$. *Next, for* $k$, *we have* $\widehat{\boldsymbol{G}}^{(\star k)}(z) = \int_0^\infty e^{-zt}\left(\boldsymbol{G}(t)\star\boldsymbol{G}^{(\star k-1)}(t)\right)dt = \widehat{\boldsymbol{G}}(z)\widehat{\boldsymbol{G}}^{(\star k-1)}(z)$, *which is* $\left(\frac{1}{z+\omega}\boldsymbol{A}\right)\left(\frac{\boldsymbol{A}^{k-1}}{(z+\omega)^{k-1}}\cdot\frac{1}{z}\right) = \frac{1}{z}\cdot\frac{\boldsymbol{A}^k}{(z+\omega)^k}$, *and completes the proof.* ■

## C.3 Proof of Theorem 6.3

**Theorem 6.3** $\boldsymbol{\mu}(t) = \boldsymbol{\Psi}(t)\boldsymbol{\lambda}^{(0)} = \left(e^{(\boldsymbol{A}-\omega\boldsymbol{I})t} + \omega(\boldsymbol{A}-\omega\boldsymbol{I})^{-1}(e^{(\boldsymbol{A}-\omega\boldsymbol{I})t} - \boldsymbol{I})\right)\boldsymbol{\lambda}^{(0)}$.

**Proof** *We first compute the Laplace transform* $\widehat{\boldsymbol{\Psi}}(z) := \int_0^\infty e^{-zt}\boldsymbol{\Psi}(t)\,dt$. *Using lemma 6.2, we have*

$$\widehat{\boldsymbol{\Psi}}(z) = \frac{1}{z}\sum_{i=0}^{\infty}\frac{\boldsymbol{A}^i}{(z+w)^i} = \frac{(z+w)}{z}\sum_{i=0}^{\infty}\frac{\boldsymbol{A}^{i+1}}{(z+w)^i}$$

*Let* $\widehat{\boldsymbol{F}}(z) := \sum_{i=0}^{\infty}\frac{\boldsymbol{A}^i}{z^{i+1}}$ *and its inverse Laplace transform be* $\boldsymbol{F}(t) = \int_0^\infty e^{zt}\widehat{\boldsymbol{F}}(z)\,dz = \sum_{i=0}^{\infty}\frac{(\boldsymbol{A}t)^i}{i!} = e^{\boldsymbol{A}t}$, *where* $e^{\boldsymbol{A}t}$ *is a matrix exponential. Then, it is easy to see that* $\widehat{\boldsymbol{\Psi}}(z) = \frac{(z+w)}{z}\widehat{\boldsymbol{F}}(z+w) = \widehat{\boldsymbol{F}}(z+w) + \frac{w}{z}\widehat{\boldsymbol{F}}(z+w)$. *Finally, we perform inverse Laplace transform for* $\widehat{\boldsymbol{\Psi}}(z)$, *and obtain* $\boldsymbol{\Psi}(t) = e^{(\boldsymbol{A}-\omega\boldsymbol{I})t} + \omega\int_0^t e^{(\boldsymbol{A}-\omega\boldsymbol{I})s}ds = e^{(\boldsymbol{A}-\omega\boldsymbol{I})t} + \omega(\boldsymbol{A}-\omega\boldsymbol{I})^{-1}\left(e^{(\boldsymbol{A}-\omega\boldsymbol{I})t} - \boldsymbol{I}\right)$, *where we made use of the property of Laplace transform that dividing by* $z$ *in the frequency domain is equal to an integration in time domain, and* $F(z+w) = e^{-\omega t}e^{\boldsymbol{A}t} = e^{(\boldsymbol{A}-\omega\boldsymbol{I})t}$. ■

## C.4 Proof of Corollary 6.4

**Corollary 6.4** $\boldsymbol{\mu} = (\mathbf{I}-\boldsymbol{\Gamma})^{-1}\boldsymbol{\lambda}^{(0)} = \lim_{t\to\infty}\boldsymbol{\Psi}(t)\,\boldsymbol{\lambda}^{(0)}$.

**Proof** *If the process is stationary, the spectral radius of* $\boldsymbol{\Gamma} = \frac{\boldsymbol{A}}{w}$ *is smaller than 1, which implies that all eigenvalues of* $\boldsymbol{A}$ *are smaller than* $\omega$ *in magnitude. Thus, all eigenvalues of* $\boldsymbol{A}-\omega\boldsymbol{I}$ *are negative. Let* $\boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{-1}$ *be the eigenvalue decomposition of* $\boldsymbol{A}-\omega\boldsymbol{I}$, *and all the elements (in diagonal) of* $\boldsymbol{D}$ *are negative. Then based on the property of matrix exponential, we have* $e^{(\boldsymbol{A}-\omega\boldsymbol{I})t} = \boldsymbol{P}e^{\boldsymbol{D}t}\boldsymbol{P}^{-1}$. *As we let* $t\to\infty$, *the matrix* $e^{\boldsymbol{D}t}\to\mathbf{0}$ *and hence* $e^{(\boldsymbol{A}-\omega\boldsymbol{I})t}\to\mathbf{0}$. *Thus* $\lim_{t\to\infty}\boldsymbol{\Psi}(t) = -\omega(\boldsymbol{A}-\omega\boldsymbol{I})^{-1}$, *which is equal to* $(\boldsymbol{I}-\boldsymbol{\Gamma})^{-1}$, *and completes the proof.* ■

# APPENDIX D

# SIMULATION

Ogata's thinning algorithm [124] is a simple and classic simulator using the idea of rejection sampling. Specifically, at any time $t$, we define a homogeneous Poisson process on some interval $[t, t+\tau(t)]$ with the chosen intensity function $\lambda_0(t)$ such that

$$\lambda_0(t) \geq \sup_{\tau \in [t, t+\tau(t)]} \lambda^*(\tau), \tag{D.1}$$

where $\tau(t)$ represents how far we can go from $t$ into the future and it can be infinite in many cases. Given the history $\mathcal{H}_t = \{t_i\}_{i=1}^n$, the key idea to simulate the next event $t_{n+1}$ is that we first generate a candidate event from the homogeneous Poisson process at the time $\hat{t}_{n+1}$ and then decide whether to keep it or not based on two conditions. If $\hat{t}_{n+1} > \tau(t_n)$, then no events will happen between $t_n$ and $t_n + \tau(t_n)$, so we restart from $t_n + \tau(t_n)$. Otherwise, an event might occur within the interval $[t_n, t_n + \tau(t_n)]$, and we accept the point at the time $\hat{t}_{n+1}$ with the probability $\lambda^*(\hat{t}_{n+1})/\lambda_0(t_n)$. The intuition is that we first generate points from the homogeneous Poisson process with large intensities $\lambda_0(t)$ (which is easy and straightforward). These generated points might be too many for the true point process, so we thin out these points later according to $\lambda^*(t)$. Algorithm D.1 gives the formal procedure, where $\text{Exp}(\alpha)$ is the exponential distribution with the expectation $1/\alpha$, and $\text{Uniform}([0, 1])$ is the uniform distribution defined from 0 to 1.

Algorithm D.1 can be generalized to the multidimensional setting. Let $\lambda_d^*(t)$ be the conditional intensity function on the $d$-th dimension where $d = 1 \dots D$, and define

$$\lambda_0^D(t) \geq \sup_{\tau \in [t, t+\tau(t)]} \sum_{d=1}^{D} \lambda_d^*(\tau). \tag{D.2}$$

**Algorithm D.1:** Simulating One-Dimensional Point Process

**Input**: $\lambda^*(t)$, $\lambda_0(t)$ and $T$.
**Output**: $\{t_i\}_{i=1}^n$

Set $n = 0$, $t = 0$;
**while** $t < T$ **do**
  Generate $s \sim \text{Exp}(\lambda_0(t))$;
  Generate $U \sim \text{Uniform}([0, 1])$;
  **if** $s > \tau(t)$ **then** $t = t + \tau(t)$;
  **else if** $t + s > T$ *or* $U > \frac{\lambda^*(t+s)}{\lambda_0(t)}$ **then** $t = t + s$ ;
  **else** $n = n + 1$, $t = t + s$, $t_n = t$ ;
**end**
**return** $\{t_i\}_{i=1}^n$

The procedure can be then rewritten in Algorithm D.2. Compared to the one-dimensional case, after we decide to accept the candidate of the next event, we need to further assign this event to the dimension that most likely contributes to its occurrence in line 10 of Algorithm D.2.

**Algorithm D.2:** Simulating Multi-Dimensional Point Process

**Input**: $\{\lambda_d^*(t)\}_{d=1}^D$, $\lambda_0^D(t)$ and $T$.
**Output**: $\left\{t_i^d\right\}_{i=1\ldots n}^{d=1\ldots D}$

1 Set $n = 0$, $t = 0$;
2 **while** $t < T$ **do**
3  Generate $s \sim \text{Exp}(\lambda_0^D(t))$;
4  Generate $U \sim \text{Uniform}([0, 1])$;
5  $t = t + s$;
6  **if** $s > \tau(t)$ **then** $t = t + \tau(t)$;
7  **else if** $t + s > T$ *or* $U > \frac{\sum_{d=1}^D \lambda_d^*(t+s)}{\lambda_0^D(t)}$ **then** $t = t + s$ ;
8  **else**
9  $n = n + 1$, $t = t + s$;
10  Pick the $d$-th dimension with the probability $\frac{\lambda_d^*(t)}{\sum_{d=1}^D \lambda_d^*(t)}$;
11  $t_n^d = t$;
12  **end**
13 **end**
14 **return** $\left\{t_i^d\right\}_{i=1\ldots n}^{d=1\ldots D}$

# REFERENCES


[1] Aalen, O., Borgan, O., and Gjessing, H., *Survival and event history analysis: a process point of view*. Springer, 2008.

[2] Adar, E. and Adamic, L. A., "Tracking information epidemics in blogspace.," in *Web Intelligence*, pp. 207–214, 2005.

[3] Ahmed, A., Eisenstein, J., Ho, Q., Xing, E. P., Smola, A. J., and Teo, C. H., "The topic-cluster model," in *Artificial Intelligence and Statistics AISTATS*, 2011. submitted.

[4] Ahmed, A., Ho, Q., Eisenstein, J., Xing, E., Smola, A., and Teo, C., "Unified analysis of streaming news," in *Proceedings of WWW*, (Hyderabad, India), IW3C2, Sheridan Printing, 2011.

[5] Ahmed, A. and Xing, E. P., "Recovering time-varying networks of dependencies in social and biological studies," *Proc. Natl. Acad. Sci. USA*, vol. 106, no. 29, pp. 11878–11883, 2009.

[6] Ahmed, A. and Xing, E., "Dynamic non-parametric mixture models and the recurrent chinese restaurant process: with applications to evolutionary clustering," in *SDM*, pp. 219–230, SIAM, 2008.

[7] Al-Mohy, A. H. and Higham, N. J., "Computing the action of the matrix exponential, with an application to exponential integrators," *SIAM journal on scientific computing*, vol. 33, no. 2, pp. 488–511, 2011.

[8] Antoniak, C., "Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems," *Annals of Statistics*, vol. 2, pp. 1152–1174, 1974.

[9] Backstrom, L., Huttenlocher, D., Kleinberg, J., and Lan, X., "Group formation in large social networks: Membership, growth, and evolution," in *Proc. of KDD'06*, 2006.

[10] Bacry, E., Delattre, S., Hoffmann, M., and Muzy, J.-F., "Modelling microstructure noise with mutually exciting point processes," *Quantitative Finance*, vol. 13, no. 1, pp. 65–77, 2013.

[11] Bacry, E., Iuga, A., Lasnier, M., and Lehalle, C.-A., "Market impacts and the life cycle of investors orders," 2014.

[12] Bacry, E., Jaisson, T., and Muzy, J.-F., "Estimation of slowly decreasing hawkes kernels: Application to high frequency order book modelling," 2015.

[13] Baddeley, A. and Turner, R., "spatstat: An r package for analyzing spatial point patterns.," *Journal of Statistical Software*, vol. 12, no. 6, pp. 1–42, 2005.

[14] Baddeley, A. and Turner, R., "spatstat: Spatial point pattern analysis, model fitting, simulation, tests.." http://CRAN.R-project.org/package=spatstat, 2010.

[15] Baltrunas, L. and Amatriain, X., "Towards time-dependant recommendation based on implicit feedback," 2009.

[16] Baum, L. E. and Petrie, T., "Statistical inference for probabilistic functions of finite state markov chains," *The Annals of Mathematical Statistics*, vol. 37, pp. 1554–1563, 12 1966.

[17] Beck, A. and Teboulle, M., "Gradient-based algorithms with applications to signal recovery," *Convex Optimization in Signal Processing and Communications*, 2009.

[18] Begleiter, R., El-Yaniv, R., and Yona, G., "On prediction using variable order markov models.," *J. Artif. Intell. Res. (JAIR)*, vol. 22, pp. 385–421, 2004.

[19] Bishop, C., *Pattern Recognition and Machine Learning*. Springer, 2006.

[20] Blei, D., Ng, A., and Jordan, M., "Latent dirichlet allocation," in *Advances in Neural Information Processing Systems 14* (Dietterich, T. G., Becker, S., and Ghahramani, Z., eds.), (Cambridge, MA), MIT Press, 2002.

[21] Blei, D. M. and Lafferty, J. D., "Dynamic topic models," in *ICML* (Cohen, W. W. and Moore, A., eds.), vol. 148, pp. 113–120, ACM, 2006.

[22] Blei, D. and Frazier, P., "Distance dependent chinese restaurant processes," in *27th International Conference on Machine Learning ICML* (Fürnkranz, J. and Joachims, T., eds.), pp. 87–94, Omnipress, 2010.

[23] Blundell, C., Beck, J., and Heller, K. A., "Modelling reciprocating relationships with hawkes processes," in *nips*, 2012.

[24] Boyd, S. and Vandenberghe, L., *Convex Optimization*. Cambridge, England: Cambridge University Press, 2004.

[25] Cha, M., Haddadi, H., Benevenuto, F., and Gummadi, P. K., "Measuring User Influence in Twitter: The Million Follower Fallacy," in *ICWSM*, 2010.

[26] Chen, W., Lu, W., and Zhang, N., "Time-critical influence maximization in social networks with time-delayed diffusion process.," in *AAAI*, 2012.

[27] Chen, W., Wang, C., and Wang, Y., "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1029–1038, ACM, 2010.

[28] Chen, W., Wang, Y., and Yang, S., "Efficient influence maximization in social networks," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 199–208, ACM, 2009.

[29] Chen, W., Yuan, Y., and Zhang, L., "Scalable influence maximization in social networks under the linear threshold model," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pp. 88–97, IEEE, 2010.

[30] Chi, E. C. and Kolda, T. G., "On tensors, sparsity, and nonnegative factorizations," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 4, pp. 1272–1299, 2012.

[31] Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y., "On the properties of neural machine translation: Encoder-decoder approaches," *CoRR*, vol. abs/1409.1259, 2014.

[32] Cohen, E., "Learning noisy perceptrons by a perceptron in polynomial time," in *Proc. 38th Annu. IEEE Symposium on Foundations of Computer Science*, pp. 514–523, IEEE Computer Society Press, Los Alamitos, CA, 1997.

[33] Cohen, E., "Size-estimation framework with applications to transitive closure and reachability," *Journal of Computer and System Sciences*, vol. 55, no. 3, pp. 441–453, 1997.

[34] Cohen, E., Delling, D., Pajor, T., and Werneck, R. F., "Sketch-based influence maximization and computation: Scaling up with guarantees," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, (New York, NY, USA), pp. 629–638, ACM, 2014.

[35] Daley, D. and Vere-Jones, D., *An introduction to the theory of point processes: volume II: general theory and structure*, vol. 2. Springer, 2007.

[36] de Nooy, W., "Networks of action and events over time. a multilevel discrete-time event history model for longitudinal network data.," *Social Networks*, vol. 33, no. 1, pp. 31–40, 2011.

[37] Diao, Q. and Jiang, J., "Recurrent chinese restaurant process with a duration-based discount for event identification from twitter," in *SDM*, 2014.

[38] Dobson, I., Carreras, B. A., and Newman, D. E., "A branching process approximation to cascading load-dependent system failure," in *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pp. 10–pp, IEEE, 2004.

[39] Doucet, A., de Freitas, J. F., Murphy, K., and Russell, S., "Rao-blackwellised particle filtering for dynamic bayesian networks," in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-00)*

(Boutilier, C. and Goldszmidt, M., eds.), (SF, CA), pp. 176–183, Morgan Kaufmann Publishers, 2000.

[40] Doucet, A., de Freitas, N., and Gordon, N., *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[41] Du, N., Song, L., Woo, H., and Zha, H., "Uncover topic-sensitive information diffusion networks," in *Artificial Intelligence and Statistics (AISTATS)*, 2013.

[42] Du, N., Farajtabar, M., Ahmed, A., Smola, A. J., and Song, L., "Dirichlet-hawkes processes with applications to clustering continuous-time document streams," in *KDD*, ACM, 2015.

[43] Du, N., Song, L., Gomez-Rodriguez, M., and Zha, H., "Scalable influence estimation in continuous-time diffusion networks," in *Advances in Neural Information Processing Systems 26*, 2013.

[44] Du, N., Song, L., Smola, A. J., and Yuan, M., "Learning networks of heterogeneous influence," in *NIPS*, 2012.

[45] Du, N., Wang, Y., He, N., and Song, L., "Time-sensitive recommendation from recurrent user activities," in *NIPS*, 2015.

[46] Eagle, N., Pentland, A. S., and Lazer, D., "From the cover: Inferring friendship network structure by using mobile phone data," *Proceedings of the National Academy of Sciences*, vol. 106, no. 36, pp. 15274–15278, 2009.

[47] Easley, D. and Kleinberg, J., *Networks, crowds, and markets*, vol. 8. Cambridge Univ Press, 2010.

[48] Egesdal, M., Fathauer, C., Louie, K., Neuman, J., Mohler, G., and Lewis, E., "Statistical and stochastic modeling of gang rivalries in los angeles," *SIAM Undergraduate Research Online*, vol. 3, pp. 72–94, 2010.

[49] Engle, R. F. and Russell, J. R., "Autoregressive conditional duration: A new model for irregularly spaced transaction data," *Econometrica*, vol. 66, pp. 1127–1162, Sep 1998.

[50] Ferraz Costa, A., Yamaguchi, Y., Juci Machado Traina, A., Traina, Jr., C., and Faloutsos, C., "Rsc: Mining and modeling temporal activity in social media," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pp. 269–278, 2015.

[51] Finkel, J. R., Grenager, T., and Manning, C., "Incorporating non-local information into information extraction systems by gibbs sampling," in *ACL*, 2005.

[52] Friedman, N. and Goldszmidt, M., "Learning bayesian networks with local structure," *Learning in graphical models*, pp. 421–460, 1998.

[53] Friedman, N. and Koller, D., "Being bayesian about network structure: a bayesian approach to structure discovery in bayesian networks," *Machine Learning*, vol. 50, pp. 95–126, 2003.

[54] Goldenberg, A., Zheng, A. X., Fienberg, S. E., and Airoldi, E. M., "A survey of statistical network models," 2009.

[55] Golub, G. H. and Van Loan, C. F., *Matrix computations*, vol. 3. JHU Press, 2012.

[56] Gomez-Rodriguez, M., Balduzzi, D., and Schölkopf, B., "Uncovering the temporal dynamics of diffusion networks," in *Proceedings of the International Conference on Machine Learning*, 2011.

[57] Gomez-Rodriguez, M., Gummadi, K., and Schoelkopf, B., "Quantifying Information Overload in Social Media and its Impact on Social Contagions," in *ICWSM*, 2014.

[58] Gomez-Rodriguez, M., Leskovec, J., and Krause, A., "Inferring networks of diffusion and influence," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1019–1028, ACM, 2010.

[59] Gomez-Rodriguez, M., Leskovec, J., and Schölkopf, B., "Structure and Dynamics of Information Pathways in On-line Media," in *WSDM*, 2013.

[60] Goyal, A., Bonchi, F., and Lakshmanan, L. V. S., "A data-based approach to social influence maximization," *Proc. VLDB Endow.*, vol. 5, 2011.

[61] Grant, S. and Betts, B., "Encouraging user behaviour with achievements: an empirical study," in *Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on*, pp. 65–68, IEEE, 2013.

[62] Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., , and Schmidhuber, J., "A novel connectionist system for uncon- strained handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 855–868, 2009.

[63] Griffiths, T. and Ghahramani, Z., "The indian buffet process: An introduction and review," *Journal of Machine Learning Research*, vol. 12, pp. 1185–1224, 2011.

[64] Halpin, P. and Boeck, P. D., "Modelling dyadic interaction with hawkes processes," *Psychometrika*, vol. 78, 2013.

[65] Hamilton, J. D., *Time Series Analysis*. Princeton University Press, 1994.

[66] Harris, T. E., *The theory of branching processes*. Courier Dover Publications, 2002.

[67] Harte, D., "Documentation for the statistical seismology library," 1998.

[68] Harte, D., "Ptprocess: An r package for modelling marked point processes indexed by time," *Journal of Statistical Software*, vol. 35, 2010.

[69] Hasan, M. A. and Zaki, M. J., "A survey of link prediction in social networks.," in *Social Network Data Analytics* (Aggarwal, C. C., ed.), pp. 243–275, Springer, 2011.

[70] Hastie, T., Tibshirani, R., and Friedman, J., *The Elements of Statistical Learning*. New York: Springer, 2001.

[71] Hawkes, A. G., "Point spectra of some mutually exciting point processes," *Journal of the Royal Statistical Society Series B*, vol. 33, pp. 438–443, 1971.

[72] Hawkes, A. G., "Spectra of some self-exciting and mutually exciting point processes," *Biometrika*, vol. 58, no. 1, pp. 83–90, 1971.

[73] Hawkes, A. G. and Oakes, D., "A cluster process representation of a self-exciting process," *Journal of Applied Probability*, pp. 493–503, 1974.

[74] Ho, Q., Song, L., and Xing, E., "Evolving cluster mixed-membership blockmodel for time-varying networks," in *Proc. Intl. Conference on Artificial Intelligence and Statistics*, pp. 342–350, 2011.

[75] Hochreiter, S. and Schmidhuber, J., "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[76] Hunter, D. R., Krivitsky, P. N., and Schweinberger, M., "Computational statistical methods for social network models," *Journal of Computational and Graphical Statistics*, vol. 21, no. 4, pp. 856–882, 2012.

[77] Ilya Sutskever, O. V. and Le, Q. V., "Sequence to sequence learning with neural networks," 2014.

[78] Ioffe, S. and Szegedy, C., "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[79] Isham, V. and Westcott, M., "A self-correcting pint process," *Advances in Applied Probability*, vol. 37, pp. 629–646, 1979.

[80] Iwata, T., Shah, A., and Ghahramani, Z., "Discovering latent influence in online social activities via shared cascade poisson processes," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 266–274, ACM, 2013.

[81] Janssen, J. and Limnios, N., *Semi-Markov Models and Applications*. Kluwer Academic, 1999.

[82] Kalman, R., "A new approach to linear filtering and prediction problems," vol. 82, pp. 35–45, 1960.

[83] Kapoor, K., Sun, M., Srivastava, J., and Ye, T., "A hazard based approach to user return time prediction," in *Knowledge discovery and data mining*, pp. 1719–1728, ACM, 2014.

[84] Kapoor, K., Subbian, K., Srivastava, J., and Schrater, P., "Just in time recommendations: Modeling the dynamics of boredom in activity streams," in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pp. 233–242, 2015.

[85] Karatzoglou, A., Amatriain, X., Baltrunas, L., and Oliver, N., "Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering," in *Recsys*, pp. 79–86, ACM, 2010.

[86] Kempe, D., Kleinberg, J., and Tardos, É., "Maximizing the spread of influence through a social network," in *SIGKDD*, pp. 137–146, ACM, 2003.

[87] Kingman, J., "On doubly stochastic poisson processes," *Mathematical Proceedings of the Cambridge Philosophical Society*, pp. 923–930, 1964.

[88] Kingman, J. F. C., *Poisson processes*, vol. 3. Oxford university press, 1992.

[89] Koenigstein, N., Dror, G., and Koren, Y., "Yahoo! music recommendations: Modeling music ratings with temporal dynamics and item taxonomy," in *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pp. 165–172, 2011.

[90] Kolar, M., Song, L., Ahmed, A., and Xing, E. P., "Estimating time-varying networks," *Ann. Appl. Statist.*, vol. 4, no. 1, pp. 94–123, 2010.

[91] Kolar, M., Song, L., and Xing, E. P., "Sparsistent learning of varying-coefficient models with structural changes," in *Advances in Neural Information Processing Systems*, pp. 1006–1014, 2009.

[92] Koller, D. and Friedman, N., *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[93] Koren, Y., "Collaborative filtering with temporal dynamics," in *KDD*, 2009.

[94] Krause, A., *Ph.D. Thesis*. CMU, 2008.

[95] Lan, G., "An optimal method for stochastic composite optimization," *Mathematical Programming*, 2012.

[96] Lan, G., "The complexity of large-scale convex programming under a linear optimization oracle," *arXiv preprint arxiv:1309.5550v2*, 2014.

[97] Lawless, J. F., *Statistical Models and Methods for Lifetime Data*. Wiley-Interscience, 2002.

[98] LeBras, R., Dilkina, B. N., Xue, Y., Gomes, C. P., McKelvey, K. S., Schwartz, M. K., and Montgomery, C. A., "Robust network design for multispecies conservation.," in *AAAI*, 2013.

[99] Leskovec, J., Backstrom, L., and Kleinberg, J., "Meme-tracking and the dynamics of the news cycle," in *KDD*, pp. 497–506, ACM, 2009.

[100] Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., and Glance, N., "Cost-effective outbreak detection in networks," in *Conference on Knowledge Discovery and Data Mining* (Berkhin, P., Caruana, R., and Wu, X., eds.), pp. 420–429, ACM, 2007.

[101] Leskovec, J., Lang, K., and Mahoney, M., "Empirical comparison of algorithms for network community detection," in *Conference on World Wide Web WWW*, (New York, NY, USA), pp. 631–640, ACM, 2010.

[102] Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C., and Ghahramani, Z., "Kronecker graphs: An approach to modeling networks," *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 985–1042, 2010.

[103] Leskovec, J., Singh, A., and Kleinberg, J. M., "Patterns of influence in a recommendation network," in *PAKDD*, pp. 380–389, 2006.

[104] Liben-Nowell, D. and Kleinberg, J., "The link prediction problem for social networks," in *Conference on Knowledge and Information Management CKIM*, (New York, NY, USA), pp. 556–559, ACM, 2003.

[105] Linderman, S. W. and Adams, R. P., "Discovering latent network structure in point process data," *arXiv preprint arXiv:1402.0914*, 2014.

[106] Liniger, T. J., *Multivariate Hawkes Processes*. PhD thesis, Swiss Federal Institute of Technology Zurich, 2009.

[107] Lozano, A. C. and Sindhwani, V., "Block variable selection in multivariate regression and high-dimensional causal inference," in *NIPS*, pp. 1486–1494, 2010.

[108] Lu, L. and Zhou, T., "Link prediction in complex networks: A survey," *Physica A*, vol. 390, no. 6, 2011.

[109] Malmgren, R. D., Hofman, J. M., Amaral, L. A. N., and Watts, D. J., "Characterizing individual communication patterns.," in *KDD*, pp. 607–616, ACM, 2009.

[110] Malmgren, R. D., Stouffer, D. B., Motter, A. E., and Amaral, L. A. N., "A Poissonian explanation for heavy tails in e-mail communication," *Proceedings of the National Academy of Sciences*, vol. 105, no. 47, pp. 18153–18158, 2008.

[111] Marsan, D. and Lengline, O., "Extending earthquakes' reach through cascading," *Science*, vol. 319, no. 5866, pp. 1076–1079, 2008.

[112] Masuda, N., Takaguchi, T., Sato, N., and Yano, K., "Self-exciting point process modeling of conversation event sequences," *Temporal Networks*, pp. 245–264, 2013.

[113] Matsubara, Y., Sakurai, Y., and Faloutsos, C., "The web as a jungle: Non-linear dynamical systems for co-evolving online activities," in *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, (New York, NY, USA), pp. 721–731, ACM, 2015.

[114] Matsubara, Y., Sakurai, Y., Faloutsos, C., Iwata, T., and Yoshikawa, M., "Fast mining and forecasting of complex time-stamped events," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, (New York, NY, USA), pp. 271–279, ACM, 2012.

[115] Matsubara, Y., Sakurai, Y., Prakash, B. A., Li, L., and Faloutsos, C., "Rise and fall patterns of information diffusion: Model and implications," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, (New York, NY, USA), pp. 6–14, 2012.

[116] Matsubara, Y., Sakurai, Y., van Panhuis, W. G., and Faloutsos, C., "Funnel: Automatic mining of spatially coevolving epidemics," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, (New York, NY, USA), pp. 105–114, ACM, 2014.

[117] Mohler, G. O., Short, M. B., Brantingham, P. J., Schoenberg, F. P., and Tita, G. E., "Self-exciting point process modeling of crime," *Journal of the American Statistical Association*, no. 493, pp. 100–108, 2011.

[118] Mohler, G., "Modeling and estimation of multi-source clustering in crime and security data," *The Annals of Applied Statistics*, vol. 7, no. 3, pp. 1525–1539, 2013.

[119] Murphy, K. P., *Machine learning: a probabilistic perspective*. The MIT Press, 2012.

[120] Myers, S. A. and Leskovec, J., "On the convexity of latent social network inference," in *NIPS*, pp. 1741–1749, 2010.

[121] Nemhauser, G., Wolsey, L., and Fisher, M., "An analysis of approximations for maximizing submodular set functions," *Mathematical Programming*, vol. 14, no. 1, 1978.

[122] Netrapalli, P. and Sanghavi, S., "Learning the graph of epidemic cascades," in *SIGMETRICS/PERFORMANCE*, pp. 211–222, ACM, 2012.

[123] Nodelman, U., Shelton, C. R., and Koller, D., "Continuous time bayesian networks," *CoRR*, vol. abs/1301.0591, 2013.

[124] Ogata, Y., "On lewis' simulation method for point processes," *Information Theory, IEEE Transactions on*, vol. 27, no. 1, pp. 23–31, 1981.

[125] Ogata, Y., "Space-time point-process models for earthquake occurrences," *Annals of the Institute of Statistical Mathematics*, vol. 50, no. 2, pp. 379–402, 1998.

[126] Ouyang, H., He, N., Tran, L. Q., and Gray, A., "Stochastic alternating direction method of multipliers," in *ICML*, 2013.

[127] Papalexakis, E. E., Dumitras, T., Chau, D. H., Prakash, B. A., and Faloutsos, C., "Spatio-temporal mining of software adoption & penetration," in *Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 878–885, 2013.

[128] Preeti Bhargava, Thomas Phan, J. Z. J. L., "Who, what, when, and where: Multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data," in *WWW*, 2015.

[129] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes in C: The Art of Scientific Computing (2nd ed.)*. Cambridge: Cambridge University Press, 1992. ISBN 0-521-43108-5.

[130] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes in C. The Art of Scientific Computation*. Cambridge, UK: Cambridge University Press, 1994.

[131] Rachel, H., Erik, L., and Andrea, L. B., "An estimate and score algorithm for simultaneous parameter estimation and reconstruction of incomplete data on social networks," *Security Informatics*, vol. 2, 2013.

[132] Rai, P., Wang, Y., Guo, S., Chen, G., Dunson, D., and Carin, L., "Scalable bayesian low-rank decomposition of incomplete multiway tensors," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1800–1808, 2014.

[133] Rakotomamonjy, A., Bach, F., Canu, S., Grandvalet, Y., and others, "Simplemkl," *Journal of Machine Learning Research*, vol. 9, pp. 2491–2521, 2008.

[134] Rasmussen, J. G., "Temporal point processes: the conditional intensity function." http://people.math.aau.dk/~jgr/teaching/punktproc11/tpp.pdf, 2009.

[135] Rasmussen, J. G., "Bayesian inference for hawkes processes," *Methodology and Computing in Applied Probability*, vol. 15, no. 3, pp. 623–642, 2013.

[136] Ravikumar, P., Wainwright, M. J., and Lafferty, J., "High-dimensional Ising model selection using $\ell_1$-regularized logistic regression," *Annals of Statistics*, vol. 38, no. 3, pp. 1287–1319, 2010.

[137] Rendle, S., "Time-Variant Factorization Models Context-Aware Ranking with Factorization Models," vol. 330 of *Studies in Computational Intelligence*, ch. 9, pp. 137–153, 2011.

[138] Richardson, M. and Domingos, P., "Mining knowledge-sharing sites for viral marketing," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 61–70, ACM, 2002.

[139] Rodriguez, M. and Schölkopf, B., "Influence maximization in continuous time diffusion networks," in *Proceedings of the International Conference on Machine Learning*, 2012.

[140] Rowlingson, B. and Diggle, P., "Splancs: spatial point pattern analysis code in s-plus," *Computers and Geosciences*, vol. 19, pp. 627–655, 2005.

[141] Saad, Y. and Schultz, M. H., "Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on scientific and statistical computing*, vol. 7, no. 3, pp. 856–869, 1986.

[142] Sastry, S., "Some np-complete problems in linear algebra," *Honors Projects*, 1990.

[143] Schmidt, M., van den Berg, E., Friedlander, M. P., and Murphy, K., "Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm," in *Proceedings of The Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS) 2009* (van Dyk, D. and Welling, M., eds.), vol. 5, (Clearwater Beach, Florida), pp. 456–463, April 2009.

[144] Schölkopf, B. and Smola, A. J., *Learning with Kernels*. Cambridge, MA: MIT Press, 2002.

[145] Shawe-Taylor, J. and Cristianini, N., *Kernel Methods for Pattern Analysis*. Cambridge, UK: Cambridge University Press, 2004.

[146] Short, M., Mohler, G., Brantingham, P., and Tita, G., "Gang rivalry dynamics via coupled point process networks," *Discrete and Continuous Dynamical Systems Series B*, vol. 19, pp. 1459–1477, 2014.

[147] Siegelmann, H. T. and Sontag, E. D., "Turing computability with neural nets," *Applied Mathematics Letters*, vol. 4, pp. 77–80, 1991.

[148] Snijders, T. A. B., "Statistical methods for network dynamics," in *Proc of the Scientific Meeting of the Italian Statistical Society*, 2006.

[149] Song, L., Kolar, M., and Xing, E. P., "Keller: estimating time-varying interactions between genes," *Bioinformatics*, vol. 25, no. 12, p. i128, 2009.

[150] Song, L., Kolar, M., and Xing, E. P., "Time-varying dynamic bayesian networks," in *Neural Information Processing Systems*, pp. 1732–1740, 2009.

[151] Sriram, S. and Daniel, N. B., "Fast graph structure learning from unlabeled data for outbreak detection," *Emerging Health Threats Journal*, vol. 4, 2011.

[152] Stomakhin, A., Short, M. B., and Bertozzi, A. L., "Reconstruction of missing data in social networks based on temporal patterns of interactions," *Inverse Problems*, vol. 27, 2011.

[153] Suen, C., Huang, S., Eksombatchai, C., Sosic, R., and Leskovec, J., "Nifty: A system for large scale information flow tracking and clustering," in *WWW*, 2013.

[154] Sutskever, I., Martens, J., Dahl, G. E., and Hinton, G. E., "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)* (Dasgupta, S. and Mcallester, D., eds.), vol. 28, pp. 1139–1147, JMLR Workshop and Conference Proceedings, May 2013.

[155] Tang, Y., Shi, Y., and Xiao, X., "Influence maximization in near-linear time: A martingale approach," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, (New York, NY, USA), pp. 1539–1554, ACM, 2015.

[156] Teh, Y. W., "A hierarchical bayesian language model based on pitman-yor processes," in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pp. 985–992, Association for Computational Linguistics, 2006.

[157] Toke, I. M., "An introduction to hawkes processes with applications to finance." http://lamp.ecp.fr/MAS/fiQuant/ioane_files/HawkesCourseSlides.pdf, 2011.

[158] Tseng, P. and Mangasarian, C. O. L., "Convergence of a block coordinate descent method for nondifferentiable minimization," *J. Optim Theory Appl*, pp. 475–494, 2001.

[159] Valera, I., Gomez-Rodriguez, M., and Gummadi, K., "Modeling adoption of competing products and conventions in social media," *arXiv preprint arXiv:1406.0516*, 2014.

[160] Vaz de Melo, P. O. S., Faloutsos, C., and Loureiro, A. F. A., "Human Dynamics in Large Communication Networks," in *SIAM Conference on Data Mining (SDM)*, pp. 879–968, 2011.

[161] Vaz de Melo, P. O. S., Faloutsos, C., Assunção, R., and Loureiro, A., "The self-feeding process: A unifying model for communication dynamics in the web," in *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, (New York, NY, USA), pp. 1319–1330, ACM, 2013.

[162] Veen, A. and Schoenberg, F. P., "Estimation of space–time branching process models in seismology using an em–type algorithm," *Journal of the American Statistical Association*, vol. 103, no. 482, pp. 614–624, 2008.

[163] Vinyals, O., Toshev, A., Bengio, S., , and Erhan, D., "Show and tell: A neural image caption generator," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3156–3164, 2015.

[164] Wang, L., Ermon, S., and Hopcroft, J. E., "Feature-enhanced probabilistic models for diffusion network inference," in *ECML/PKDD (2)*, pp. 499–514, 2012.

[165] Wang, X. and McCallum, A., "Topics over time: A non-markov continuous-time model of topical trends," in *KDD*, 2006.

[166] Watts, D. J. and Dodds, P. S., "Influentials, networks, and public opinion formation," *Journal of Consumer Research*, vol. 34, no. 4, pp. 441–458, 2007.

[167] Watts, D. J. and Strogatz, S. H., "Collective dynamics of small-world networks," *Nature*, vol. 393, pp. 440–442, June 1998.

[168] Xiong, L., Chen, X., Huang, T.-K., Schneider, J. G., and Carbonell, J. G., "Temporal collaborative filtering with bayesian probabilistic tensor factorization.," in *SDM*, pp. 211–222, SIAM, 2010.

[169] Xu, Z., Jin, R., Yang, H., King, I., and Lyu, M., "Simple and efficient multiple kernel learning by group lasso," in *Proceedings of the International Conference on Machine Learning*, pp. 1191–1198, 2010.

[170] Yang, S.-H. and Zha, H., "Mixture of mutually exciting processes for viral diffusion," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 1–9, 2013.

[171] Yi, X., Hong, L., Zhong, E., Liu, N. N., and Rajan, S., "Beyond clicks: Dwell time for personalization," in *RecSys*, 2014.

[172] Yu, A. W., Ma, W., Yu, Y., Carbonell, J. G., and Sra, S., "Efficient structured matrix rank minimization," in *NIPS*, 2014.

[173] Zaid Harchaoui, Anatoli Juditsky, A. N., "Conditional gradient algorithms for norm-regularized smooth convex optimization," *Mathematical Programming*, 2013.

[174] Zaremba, W. and Sutskever, I., "Learning to execute," *arXiv preprint arXiv:1410.4615*, 2014.

[175] Zhou, K., Zha, H., and Song, L., "Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes," in *AISTAT*, 2013.

[176] Zhou, K., Zha, H., and Song, L., "Learning triggering kernels for multi-dimensional hawkes processes," in *ICML*, 2013.

[177] Zhuang, J., Ogata, Y., and Vere-Jones, D., "Stochastic declustering of space-time earthquake occurrences," *Journal of the American Statistical Association*, vol. 97, no. 458, pp. 369–380, 2002.

# VITA

Nan Du is a Ph.D. candidate in the School of Computational Science and Engineering, College of Computing, Georgia Institute of Technology. His primary research interest is in developing sophisticated machine learning models and large-scale algorithms for statistical analysis of networked temporal/spatial dynamics arising from social networks, online media and medical informatics, which has big impacts on promoting user-engagement, optimizing advertisement allocations and providing context-aware recommendations. Nan has received several recognitions for his research, including a Best Paper Award at NIPS 2013, a Facebook Graduate Fellowship 2014-2015, Faces of Inclusive Excellence Recognition at Georgia Tech 2014, and an Annual Outstanding Graduate Assistant Award of Georgia Tech 2014-2015.